# 分组复制

本章解释了MySQL组复制以及如何安装、配置和监控组。MySQL组复制使你能够创建弹性、高可用、容错的复制拓扑结构。

组可以在具有自动主选的单主模式下运行，即每次只有一台服务器接受更新。另外，组可以部署在多主模式下，所有服务器都可以接受更新，即使它们是同时发布的。

有一个内置的组成员服务，在任何给定的时间点都能保持组的观点一致，并对所有服务器可用。服务器可以离开和加入该组，视图也会相应地更新。有时，服务器会意外地离开组，在这种情况下，故障检测机制会检测到这一点，并通知组的视图已经改变。这都是自动的。

组复制保证了数据库服务的持续可用性。然而，重要的是要理解，如果群组成员之一变得不可用，连接到该群组成员的客户端必须使用连接器、负载平衡器、路由器或某种形式的中间件重定向，或故障转移到群组中的不同服务器。组复制没有一个内置的方法来做这个。例如，见MySQL Router 8.0。

组复制是作为MySQL服务器的一个插件提供的。你可以按照本章的说明，在你想在组中的每个服务器实例上配置该插件，启动该组，并监视和管理该组。部署一组MySQL服务器实例的另一种方法是使用InnoDB Cluster。

提示

要部署多个MySQL实例，你可以使用InnoDB Cluster，它使你能够在MySQL Shell中轻松管理一组MySQL服务器实例。InnoDB Cluster将MySQL组复制包装在一个编程环境中，使你能够轻松地部署一个MySQL实例的集群，以实现高可用性。此外，InnoDB Cluster与MySQL Router无缝对接，使你的应用程序能够连接到集群，而无需编写自己的故障转移过程。然而，对于不需要高可用性的类似用例，你可以使用InnoDB ReplicaSet。MySQL Shell的安装说明可以在这里找到。

The chapter is structured as follows:

[Section 18.1, “Group Replication Background”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-background) provides an introduction to groups and how Group Replication works.

[Section 18.2, “Getting Started”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-getting-started) explains how to configure multiple MySQL Server instances to create a group.

[Section 18.3, “Requirements and Limitations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-requirements-and-limitations) explains architecture and setup requirements and limitations for Group Replication.

[Section 18.4, “Monitoring Group Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-monitoring) explains how to monitor a group.

[Section 18.5, “Group Replication Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-operations) explains how to work with a group.

[Section 18.6, “Group Replication Security”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-security) explains how to secure a group.

[Section 18.7, “Group Replication Performance”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-performance) explains how to fine tune performance for a group.

[Section 18.8, “Upgrading Group Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-upgrade) explains how to upgrade a group.

[Section 18.9, “Group Replication System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-options) is a reference for the system variables specific to Group Replication.

[Section 18.10, “Frequently Asked Questions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-frequently-asked-questions) provides answers to some technical questions about deploying and operating Group Replication.

## 18.1 Group Replication Background

[18.1.1 Replication Technologies](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-replication-technologies)

[18.1.2 Group Replication Use Cases](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-use-cases)

[18.1.3 Multi-Primary and Single-Primary Modes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-deploying-in-multi-primary-or-single-primary-mode)

[18.1.4 Group Replication Services](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-details)

[18.1.5 Group Replication Plugin Architecture](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-plugin-architecture)

This section provides background information on MySQL Group Replication.

The most common way to create a fault-tolerant system is to resort to making components redundant, in other words the component can be removed and the system should continue to operate as expected. This creates a set of challenges that raise complexity of such systems to a whole different level. Specifically, replicated databases have to deal with the fact that they require maintenance and administration of several servers instead of just one. Moreover, as servers are cooperating together to create the group several other classic distributed systems problems have to be dealt with, such as network partitioning or split brain scenarios.

Therefore, the ultimate challenge is to fuse the logic of the database and data replication with the logic of having several servers coordinated in a consistent and simple way. In other words, to have multiple servers agreeing on the state of the system and the data on each and every change that the system goes through. This can be summarized as having servers reaching agreement on each database state transition, so that they all progress as one single database or alternatively that they eventually converge to the same state. Meaning that they need to operate as a (distributed) state machine.

MySQL Group Replication provides distributed state machine replication with strong coordination between servers. Servers coordinate themselves automatically when they are part of the same group. The group can operate in a single-primary mode with automatic primary election, where only one server accepts updates at a time. Alternatively, for more advanced users the group can be deployed in multi-primary mode, where all servers can accept updates, even if they are issued concurrently. This power comes at the expense of applications having to work around the limitations imposed by such deployments.

There is a built-in group membership service that keeps the view of the group consistent and available for all servers at any given point in time. Servers can leave and join the group and the view is updated accordingly. Sometimes servers can leave the group unexpectedly, in which case the failure detection mechanism detects this and notifies the group that the view has changed. This is all automatic.

For a transaction to commit, the majority of the group have to agree on the order of a given transaction in the global sequence of transactions. Deciding to commit or abort a transaction is done by each server individually, but all servers make the same decision. If there is a network partition, resulting in a split where members are unable to reach agreement, then the system does not progress until this issue is resolved. Hence there is also a built-in, automatic, split-brain protection mechanism.

All of this is powered by the provided Group Communication System (GCS) protocols. These provide a failure detection mechanism, a group membership service, and safe and completely ordered message delivery. All these properties are key to creating a system which ensures that data is consistently replicated across the group of servers. At the very core of this technology lies an implementation of the Paxos algorithm. It acts as the group communication engine.

### 18.1.1 Replication Technologies

[18.1.1.1 Source to Replica Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-primary-secondary-replication)

[18.1.1.2 Group Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-summary)

Before getting into the details of MySQL Group Replication, this section introduces some background concepts and an overview of how things work. This provides some context to help understand what is required for Group Replication and what the differences are between classic asynchronous MySQL Replication and Group Replication.

#### 18.1.1.1 Source to Replica Replication

Traditional MySQL [Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html) provides a simple source to replica approach to replication. The source is the primary, and there are one or more replicas, which are secondaries. The source applies transactions, commits them and then they are later (thus asynchronously) sent to the replicas to be either re-executed (in statement-based replication) or applied (in row-based replication). It is a shared-nothing system, where all servers have a full copy of the data by default.

**Figure 18.1 MySQL Asynchronous Replication**

There is also semisynchronous replication, which adds one synchronization step to the protocol. This means that the primary waits, at apply time, for the secondary to acknowledge that it has received the transaction. Only then does the primary resume the commit operation.

**Figure 18.2 MySQL Semisynchronous Replication**

In the two pictures there is a diagram of the classic asynchronous MySQL Replication protocol (and its semisynchronous variant as well). The arrows between the different instances represent messages exchanged between servers or messages exchanged between servers and the client application.

#### 18.1.1.2 Group Replication

Group Replication is a technique that can be used to implement fault-tolerant systems. The replication group is a set of servers that each have their own entire copy of the data (a shared-nothing replication scheme), and interact with each other through message passing. The communication layer provides a set of guarantees such as atomic message and total order message delivery. These are very powerful properties that translate into very useful abstractions that one can resort to build more advanced database replication solutions.

MySQL Group Replication builds on top of such properties and abstractions and implements a multi-source update everywhere replication protocol. A replication group is formed by multiple servers and each server in the group may execute transactions independently at any time. However, all read-write transactions commit only after they have been approved by the group. In other words, for any read-write transaction the group needs to decide whether it commits or not, so the commit operation is not a unilateral decision from the originating server. Read-only transactions need no coordination within the group and commit immediately.

When a read-write transaction is ready to commit at the originating server, the server atomically broadcasts the write values (the rows that were changed) and the corresponding write set (the unique identifiers of the rows that were updated). Because the transaction is sent through an atomic broadcast, either all servers in the group receive the transaction or none do. If they receive it, then they all receive it in the same order with respect to other transactions that were sent before. All servers therefore receive the same set of transactions in the same order, and a global total order is established for the transactions.

However, there may be conflicts between transactions that execute concurrently on different servers. Such conflicts are detected by inspecting and comparing the write sets of two different and concurrent transactions, in a process called certification. During certification, conflict detection is carried out at row level: if two concurrent transactions, that executed on different servers, update the same row, then there is a conflict. The conflict resolution procedure states that the transaction that was ordered first commits on all servers, and the transaction ordered second aborts, and is therefore rolled back on the originating server and dropped by the other servers in the group. For example, if t1 and t2 execute concurrently at different sites, both changing the same row, and t2 is ordered before t1, then t2 wins the conflict and t1 is rolled back. This is in fact a distributed first commit wins rule. Note that if two transactions are bound to conflict more often than not, then it is a good practice to start them on the same server, where they have a chance to synchronize on the local lock manager instead of being rolled back as a result of certification.

For applying and externalizing the certified transactions, Group Replication permits servers to deviate from the agreed order of the transactions if this does not break consistency and validity. Group Replication is an eventual consistency system, meaning that as soon as the incoming traffic slows down or stops, all group members have the same data content. While traffic is flowing, transactions can be externalized in a slightly different order, or externalized on some members before the others. For example, in multi-primary mode, a local transaction might be externalized immediately following certification, although a remote transaction that is earlier in the global order has not yet been applied. This is permitted when the certification process has established that there is no conflict between the transactions. In single-primary mode, on the primary server, there is a small chance that concurrent, non-conflicting local transactions might be committed and externalized in a different order from the global order agreed by Group Replication. On the secondaries, which do not accept writes from clients, transactions are always committed and externalized in the agreed order.

The following figure depicts the MySQL Group Replication protocol and by comparing it to MySQL Replication (or even MySQL semisynchronous replication) you can see some differences. Some underlying consensus and Paxos related messages are missing from this picture for the sake of clarity.

**Figure 18.3 MySQL Group Replication Protocol**

### 18.1.2 Group Replication Use Cases

Group Replication enables you to create fault-tolerant systems with redundancy by replicating the system state to a set of servers. Even if some of the servers subsequently fail, as long it is not all or a majority, the system is still available. Depending on the number of servers which fail the group might have degraded performance or scalability, but it is still available. Server failures are isolated and independent. They are tracked by a group membership service which relies on a distributed failure detector that is able to signal when any servers leave the group, either voluntarily or due to an unexpected halt. There is a distributed recovery procedure to ensure that when servers join the group they are brought up to date automatically. There is no need for server failover, and the multi-source update everywhere nature ensures that even updates are not blocked in the event of a single server failure. To summarize, MySQL Group Replication guarantees that the database service is continuously available.

It is important to understand that although the database service is available, in the event of an unexpected server exit, those clients connected to it must be redirected, or failed over, to a different server. This is not something Group Replication attempts to resolve. A connector, load balancer, router, or some form of middleware are more suitable to deal with this issue. For example see [MySQL Router 8.0](https://dev.mysql.com/doc/mysql-router/8.0/en/).

To summarize, MySQL Group Replication provides a highly available, highly elastic, dependable MySQL service.

Tip

To deploy multiple instances of MySQL, you can use [InnoDB Cluster](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-cluster.html) which enables you to easily administer a group of MySQL server instances in [MySQL Shell](https://dev.mysql.com/doc/mysql-shell/8.0/en/). InnoDB Cluster wraps MySQL Group Replication in a programmatic environment that enables you easily deploy a cluster of MySQL instances to achieve high availability. In addition, InnoDB Cluster interfaces seamlessly with [MySQL Router](https://dev.mysql.com/doc/mysql-router/8.0/en/), which enables your applications to connect to the cluster without writing your own failover process. For similar use cases that do not require high availability, however, you can use [InnoDB ReplicaSet](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-replicaset.html). Installation instructions for MySQL Shell can be found [here](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-install.html).

#### Example Use Cases

The following examples are typical use cases for Group Replication.

Elastic Replication - Environments that require a very fluid replication infrastructure, where the number of servers has to grow or shrink dynamically and with as few side-effects as possible. For instance, database services for the cloud.

Highly Available Shards - Sharding is a popular approach to achieve write scale-out. Use MySQL Group Replication to implement highly available shards, where each shard maps to a replication group.

Alternative to asynchronous Source-Replica replication - In certain situations, using a single source server makes it a single point of contention. Writing to an entire group may prove more scalable under certain circumstances.

Autonomic Systems - Additionally, you can deploy MySQL Group Replication purely for the automation that is built into the replication protocol (described already in this and previous chapters).

### 18.1.3 Multi-Primary and Single-Primary Modes

[18.1.3.1 Single-Primary Mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-single-primary-mode)

[18.1.3.2 Multi-Primary Mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-multi-primary-mode)

Group Replication operates either in single-primary mode or in multi-primary mode. The group's mode is a group-wide configuration setting, specified by the [**group\_replication\_single\_primary\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_single_primary_mode) system variable, which must be the same on all members. **ON** means single-primary mode, which is the default mode, and **OFF** means multi-primary mode. It is not possible to have members of the group deployed in different modes, for example one member configured in multi-primary mode while another member is in single-primary mode.

You cannot change the value of [**group\_replication\_single\_primary\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_single_primary_mode) manually while Group Replication is running. From MySQL 8.0.13, you can use the [**group\_replication\_switch\_to\_single\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-single-primary-mode) and [**group\_replication\_switch\_to\_multi\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-multi-primary-mode) UDFs to move a group from one mode to another while Group Replication is still running. These UDFs manage the process of changing the group's mode and ensure the safety and consistency of your data. In earlier releases, to change the group's mode you must stop Group Replication and change the value of [**group\_replication\_single\_primary\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_single_primary_mode) on all members. Then carry out a full reboot of the group (a bootstrap by a server with [**group\_replication\_bootstrap\_group=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group)) to implement the change to the new operating configuration. You do not need to restart the servers.

Regardless of the deployed mode, Group Replication does not handle client-side failover. That must be handled by a middleware framework such as [MySQL Router 8.0](https://dev.mysql.com/doc/mysql-router/8.0/en/), a proxy, a connector, or the application itself.

#### 18.1.3.1 Single-Primary Mode

In single-primary mode ([**group\_replication\_single\_primary\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_single_primary_mode)) the group has a single primary server that is set to read-write mode. All the other members in the group are set to read-only mode (with [**super\_read\_only=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only)). The primary is typically the first server to bootstrap the group. All other servers that join the group learn about the primary server and are automatically set to read-only mode.

In single-primary mode, Group Replication enforces that only a single server writes to the group, so compared to multi-primary mode, consistency checking can be less strict and DDL statements do not need to be handled with any extra care. The option [**group\_replication\_enforce\_update\_everywhere\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_enforce_update_everywhere_checks) enables or disables strict consistency checks for a group. When deploying in single-primary mode, or changing the group to single-primary mode, this system variable must be set to **OFF**.

The member that is designated as the primary server can change in the following ways:

If the existing primary leaves the group, whether voluntarily or unexpectedly, a new primary is elected automatically.

You can appoint a specific member as the new primary using the [**group\_replication\_set\_as\_primary()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-as-primary) UDF.

If you use the [**group\_replication\_switch\_to\_single\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-single-primary-mode) UDF to change a group that was running in multi-primary mode to run in single-primary mode, a new primary is elected automatically, or you can appoint the new primary by specifying it with the UDF.

The UDFs can only be used when all group members are running MySQL 8.0.13 or higher. When a new primary server is elected automatically or appointed manually, it is automatically set to read-write, and the other group members remain as secondaries, and as such, read-only. [Figure 18.4, “New Primary Election”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-primary-election-diagram) shows this process.

**Figure 18.4 New Primary Election**

When a new primary is elected or appointed, it might have a backlog of changes that had been applied on the old primary but have not yet been applied on this server. In this situation, until the new primary catches up with the old primary, read-write transactions might result in conflicts and be rolled back, and read-only transactions might result in stale reads. Group Replication's flow control mechanism, which minimizes the difference between fast and slow members, reduces the chances of this happening if it is activated and properly tuned. For more information on flow control, see [Section 18.7.2, “Flow Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-flow-control). From MySQL 8.0.14, you can also use the [**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) system variable to configure the group's level of transaction consistency to prevent this issue. The setting **BEFORE\_ON\_PRIMARY\_FAILOVER** (or any higher consistency level) holds new transactions on a newly elected primary until the backlog has been applied. For more information on transaction consistency, see [Section 18.5.2, “Transaction Consistency Guarantees”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-consistency-guarantees). If flow control and transaction consistency guarantees are not used for a group, it is a good practice to wait for the new primary to apply its replication-related relay log before re-routing client applications to it.

##### 18.1.3.1.1 Primary Election Algorithm

The automatic primary member election process involves each member looking at the new view of the group, ordering the potential new primary members, and choosing the member that qualifies as the most suitable. Each member makes its own decision locally, following the primary election algorithm in its MySQL Server release. Because all members must reach the same decision, members adapt their primary election algorithm if other group members are running lower MySQL Server versions, so that they have the same behavior as the member with the lowest MySQL Server version in the group.

The factors considered by members when electing a primary, in order, are as follows:

The first factor considered is which member or members are running the lowest MySQL Server version. If all group members are running MySQL 8.0.17 or higher, members are first ordered by the patch version of their release. If any members are running MySQL Server 5.7 or MySQL 8.0.16 or lower, members are first ordered by the major version of their release, and the patch version is ignored.

If more than one member is running the lowest MySQL Server version, the second factor considered is the member weight of each of those members, as specified by the [**group\_replication\_member\_weight**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_weight) system variable on the member. If any member of the group is running MySQL Server 5.7, where this system variable was not available, this factor is ignored.

The [**group\_replication\_member\_weight**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_weight) system variable specifies a number in the range 0-100. All members default to a weight of 50, so set a weight below this to lower their ordering, and a weight above it to increase their ordering. You can use this weighting function to prioritize the use of better hardware or to ensure failover to a specific member during scheduled maintenance of the primary.

If more than one member is running the lowest MySQL Server version, and more than one of those members has the highest member weight (or member weighting is being ignored), the third factor considered is the lexographical order of the generated server UUIDs of each member, as specified by the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) system variable. The member with the lowest server UUID is chosen as the primary. This factor acts as a guaranteed and predictable tie-breaker so that all group members reach the same decision if it cannot be determined by any important factors.

##### 18.1.3.1.2 Finding the Primary

To find out which server is currently the primary when deployed in single-primary mode, use the **MEMBER\_ROLE** column in the [**performance\_schema.replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table) table. For example:

mysql> **SELECT MEMBER\_HOST, MEMBER\_ROLE FROM performance\_schema.replication\_group\_members;**

+-------------------------+-------------+

| MEMBER\_HOST | MEMBER\_ROLE |

+-------------------------+-------------+

| remote1.example.com | PRIMARY |

| remote2.example.com | SECONDARY |

| remote3.example.com | SECONDARY |

+-------------------------+-------------+

**Warning**

The **group\_replication\_primary\_member** status variable has been deprecated and is scheduled to be removed in a future version.

Alternatively use the [**group\_replication\_primary\_member**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_group_replication_primary_member) status variable.

mysql> **SHOW STATUS LIKE 'group\_replication\_primary\_member'**

#### 18.1.3.2 Multi-Primary Mode

In multi-primary mode ([**group\_replication\_single\_primary\_mode=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_single_primary_mode)) no member has a special role. Any member that is compatible with the other group members is set to read-write mode when joining the group, and can process write transactions, even if they are issued concurrently.

If a member stops accepting write transactions, for example, in the event of an unexpected server exit, clients connected to it can be redirected, or failed over, to any other member that is in read-write mode. Group Replication does not handle client-side failover itself, so you need to arrange this using a middleware framework such as [MySQL Router 8.0](https://dev.mysql.com/doc/mysql-router/8.0/en/), a proxy, a connector, or the application itself. [Figure 18.5, “Client Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-multi-primary-diagram) shows how clients can reconnect to an alternative group member if a member leaves the group.

**Figure 18.5 Client Failover**

Group Replication is an eventual consistency system. This means that as soon as the incoming traffic slows down or stops, all group members have the same data content. While traffic is flowing, transactions can be externalized on some members before the others, especially if some members have less write throughput than others, creating the possibility of stale reads. In multi-primary mode, slower members can also build up an excessive backlog of transactions to certify and apply, leading to a greater risk of conflicts and certification failure. To limit these issues, you can activate and tune Group Replication's flow control mechanism to minimize the difference between fast and slow members. For more information on flow control, see [Section 18.7.2, “Flow Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-flow-control).

From MySQL 8.0.14, if you want to have a transaction consistency guarantee for every transaction in the group, you can do this using the [**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) system variable. You can choose a setting that suits the workload of your group and your priorities for data reads and writes, taking into account the performance impact of the synchronization required to increase consistency. You can also set the system variable for individual sessions to protect particularly concurrency-sensitive transactions. For more information on transaction consistency, see [Section 18.5.2, “Transaction Consistency Guarantees”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-consistency-guarantees).

##### 18.1.3.2.1 Transaction Checks

When a group is deployed in multi-primary mode, transactions are checked to ensure they are compatible with the mode. The following strict consistency checks are made when Group Replication is deployed in multi-primary mode:

If a transaction is executed under the SERIALIZABLE isolation level, then its commit fails when synchronizing itself with the group.

If a transaction executes against a table that has foreign keys with cascading constraints, then its commit fails when synchronizing itself with the group.

The checks are controlled by the [**group\_replication\_enforce\_update\_everywhere\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_enforce_update_everywhere_checks) system variable. In multi-primary mode, the system variable should normally be set to **ON**, but the checks can optionally be deactivated by setting the system variable to **OFF**. When deploying in single-primary mode, the system variable must be set to **OFF**.

##### 18.1.3.2.2 Data Definition Statements

In a Group Replication topology in multi-primary mode, care needs to be taken when executing data definition statements, also commonly known as data definition language (DDL).

MySQL 8.0 introduces support for atomic Data Definition Language (DDL) statements, where the complete DDL statement is either committed or rolled back as a single atomic transaction. However, DDL statements, atomic or otherwise, implicitly end any transaction that is active in the current session, as if you had done a [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) before executing the statement. This means that DDL statements cannot be performed within another transaction, within transaction control statements such as [**START TRANSACTION ... COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), or combined with other statements within the same transaction.

Group Replication is based on an optimistic replication paradigm, where statements are optimistically executed and rolled back later if necessary. Each server executes without securing group agreement first. Therefore, more care needs to be taken when replicating DDL statements in multi-primary mode. If you make schema changes (using DDL) and changes to the data that an object contains (using DML) for the same object, the changes need to be handled through the same server while the schema operation has not yet completed and replicated everywhere. Failure to do so can result in data inconsistency when operations are interrupted or only partially completed. If the group is deployed in single-primary mode this issue does not occur, because all changes are performed through the same server, the primary.

For details on atomic DDL support in MySQL 8.0, and the resulting changes in behavior for the replication of certain statements, see [Section 13.1.1, “Atomic Data Definition Statement Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl).

##### 18.1.3.2.3 Version Compatibility

For optimal compatibility and performance, all members of a group should run the same version of MySQL Server and therefore of Group Replication. In multi-primary mode, this is more significant because all members would normally join the group in read-write mode. If a group includes members running more than one MySQL Server version, there is a potential for some members to be incompatible with others, because they support functions others do not, or lack functions others have. To guard against this, when a new member joins (including a former member that has been upgraded and restarted), the member carries out compatibility checks against the rest of the group.

One result of these compatibility checks is particularly important in multi-primary mode. If a joining member is running a higher MySQL Server version than the lowest version that the existing group members are running, it joins the group but remains in read-only mode. (In a group that is running in single-primary mode, newly added members default to being read-only in any case.) Members running MySQL 8.0.17 or higher take into account the patch version of the release when checking their compatibility. Members running MySQL 8.0.16 or lower, or MySQL 5.7, only take into account the major version.

In a group running in multi-primary mode with members that use different MySQL Server versions, Group Replication automatically manages the read-write and read-only status of members running MySQL 8.0.17 or higher. If a member leaves the group, the members running the version that is now the lowest are automatically set to read-write mode. When you change a group that was running in single-primary mode to run in multi-primary mode, using the [**group\_replication\_switch\_to\_multi\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-multi-primary-mode) UDF, Group Replication automatically sets members to the correct mode. Members are automatically placed in read-only mode if they are running a higher MySQL server version than the lowest version present in the group, and members running the lowest version are placed in read-write mode.

For full information on version compatibility in a group and how this influences the behavior of a group during an upgrade process, see [Section 18.8.1, “Combining Different Member Versions in a Group”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-online-upgrade-combining-versions) .

### 18.1.4 Group Replication Services

[18.1.4.1 Group Membership](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-group-membership)

[18.1.4.2 Failure Detection](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-failure-detection)

[18.1.4.3 Fault-tolerance](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-fault-tolerance)

[18.1.4.4 Observability](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-observability)

This section introduces some of the services that Group Replication builds on.

#### 18.1.4.1 Group Membership

In MySQL Group Replication, a set of servers forms a replication group. A group has a name, which takes the form of a UUID. The group is dynamic and servers can leave (either voluntarily or involuntarily) and join it at any time. The group adjusts itself whenever servers join or leave.

If a server joins the group, it automatically brings itself up to date by fetching the missing state from an existing server. If a server leaves the group, for instance it was taken down for maintenance, the remaining servers notice that it has left and reconfigure the group automatically.

Group Replication has a group membership service that defines which servers are online and participating in the group. The list of online servers is referred to as a view. Every server in the group has a consistent view of which servers are the members participating actively in the group at a given moment in time.

Group members must agree not only on transaction commits, but also on which is the current view. If existing members agree that a new server should become part of the group, the group is reconfigured to integrate that server in it, which triggers a view change. If a server leaves the group, either voluntarily or not, the group dynamically rearranges its configuration and a view change is triggered.

In the case where a member leaves the group voluntarily, it first initiates a dynamic group reconfiguration, during which all members have to agree on a new view without the leaving server. However, if a member leaves the group involuntarily, for example because it has stopped unexpectedly or the network connection is down, it cannot initiate the reconfiguration. In this situation, Group Replication's failure detection mechanism recognizes after a short period of time that the member has left, and a reconfiguration of the group without the failed member is proposed. As with a member that leaves voluntarily, the reconfiguration requires agreement from the majority of servers in the group. However, if the group is not able to reach agreement, for example because it partitioned in such a way that there is no majority of servers online, the system is not able to dynamically change the configuration, and blocks to prevent a split-brain situation. This situation requires intervention from an administrator.

It is possible for a member to go offline for a short time, then attempt to rejoin the group again before the failure detection mechanism has detected its failure, and before the group has been reconfigured to remove the member. In this situation, the rejoining member forgets its previous state, but if other members send it messages that are intended for its pre-crash state, this can cause issues including possible data inconsistency. If a member in this situation participates in XCom's consensus protocol, it could potentially cause XCom to deliver different values for the same consensus round, by making a different decision before and after failure.

To counter this possibility, from MySQL 5.7.22 and in MySQL 8.0, Group Replication checks for the situation where a new incarnation of the same server is trying to join the group while its old incarnation (with the same address and port number) is still listed as a member. The new incarnation is blocked from joining the group until the old incarnation can be removed by a reconfiguration. Note that if a waiting period has been added by the [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) system variable to allow additional time for members to reconnect with the group before they are expelled, a member under suspicion can become active in the group again as its current incarnation if it reconnects to the group before the suspicion times out. When a member exceeds the expel timeout and is expelled from the group, or when Group Replication is stopped on the server by a [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) statement or a server failure, it must rejoin as a new incarnation.

#### 18.1.4.2 Failure Detection

Group Replication includes a failure detection mechanism that is able to find and report which servers are silent and as such assumed to be dead. At a high level, the failure detector is a distributed service that provides information about which servers may be dead (suspicions). Suspicions are triggered when servers go mute. When server A does not receive messages from server B during a given period, a timeout occurs and a suspicion is raised. Later if the group agrees that the suspicions are probably true, then the group decides that a given server has indeed failed. This means that the remaining members in the group take a coordinated decision to expel a given member.

If a server gets isolated from the rest of the group, then it suspects that all others have failed. Being unable to secure agreement with the group (as it cannot secure a quorum), its suspicion does not have consequences. When a server is isolated from the group in this way, it is unable to execute any local transactions.

Where the network is unstable and members frequently lose and regain connection to each other in different combinations, it is theoretically possible for a group to end up marking all its members for expulsion, after which the group would cease to exist and have to be set up again. To counter this possibility, from MySQL 8.0.20, Group Replication's Group Communication System (GCS) tracks the group members that have been marked for expulsion, and treats them as if they were in the group of suspected members when deciding if there is a majority. This ensures at least one member remains in the group and the group can continue to exist. When an expelled member has actually been removed from the group, GCS removes its record of having marked the member for expulsion, so that the member can rejoin the group if it is able to.

For information on the Group Replication system variables that you can configure to specify the responses of working group members to failure situations, and the actions taken by group members that are suspected of having failed, see [Section 18.7.6, “Responses to Failure Detection and Network Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-responses-failure).

#### 18.1.4.3 Fault-tolerance

MySQL Group Replication builds on an implementation of the Paxos distributed algorithm to provide distributed coordination between servers. As such, it requires a majority of servers to be active to reach quorum and thus make a decision. This has direct impact on the number of failures the system can tolerate without compromising itself and its overall functionality. The number of servers (n) needed to tolerate **f** failures is then **n = 2 x f + 1**.

In practice this means that to tolerate one failure the group must have three servers in it. As such if one server fails, there are still two servers to form a majority (two out of three) and allow the system to continue to make decisions automatically and progress. However, if a second server fails involuntarily, then the group (with one server left) blocks, because there is no majority to reach a decision.

The following is a small table illustrating the formula above.

| **Group Size** | **Majority** | **Instant Failures Tolerated** |
| --- | --- | --- |
| **1** | 1 | 0 |
| **2** | 2 | 0 |
| **3** | 2 | 1 |
| **4** | 3 | 1 |
| **5** | 3 | 2 |
| **6** | 4 | 2 |
| **7** | 4 | 3 |

#### 18.1.4.4 Observability

There is a lot of automation built into the Group Replication plugin. Nonetheless, you might sometimes need to understand what is happening behind the scenes. This is where the instrumentation of Group Replication and Performance Schema becomes important. The entire state of the system (including the view, conflict statistics and service states) can be queried through Performance Schema tables. The distributed nature of the replication protocol and the fact that server instances agree and thus synchronize on transactions and metadata makes it simpler to inspect the state of the group. For example, you can connect to a single server in the group and obtain both local and global information by issuing select statements on the Group Replication related Performance Schema tables. For more information, see [Section 18.4, “Monitoring Group Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-monitoring).

### 18.1.5 Group Replication Plugin Architecture

MySQL Group Replication is a MySQL plugin and it builds on the existing MySQL replication infrastructure, taking advantage of features such as the binary log, row-based logging, and global transaction identifiers. It integrates with current MySQL frameworks, such as the performance schema or plugin and service infrastructures. The following figure presents a block diagram depicting the overall architecture of MySQL Group Replication.

**Figure 18.6 Group Replication Plugin Block Diagram**

The MySQL Group Replication plugin includes a set of APIs for capture, apply, and lifecycle, which control how the plugin interacts with MySQL Server. There are interfaces to make information flow from the server to the plugin and vice versa. These interfaces isolate the MySQL Server core from the Group Replication plugin, and are mostly hooks placed in the transaction execution pipeline. In one direction, from server to the plugin, there are notifications for events such as the server starting, the server recovering, the server being ready to accept connections, and the server being about to commit a transaction. In the other direction, the plugin instructs the server to perform actions such as committing or aborting ongoing transactions, or queuing transactions in the relay log.

The next layer of the Group Replication plugin architecture is a set of components that react when a notification is routed to them. The capture component is responsible for keeping track of context related to transactions that are executing. The applier component is responsible for executing remote transactions on the database. The recovery component manages distributed recovery, and is responsible for getting a server that is joining the group up to date by selecting the donor, managing the catch up procedure and reacting to donor failures.

Continuing down the stack, the replication protocol module contains the specific logic of the replication protocol. It handles conflict detection, and receives and propagates transactions to the group.

The final two layers of the Group Replication plugin architecture are the Group Communication System (GCS) API, and an implementation of a Paxos-based group communication engine (XCom). The GCS API is a high level API that abstracts the properties required to build a replicated state machine (see [Section 18.1, “Group Replication Background”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-background)). It therefore decouples the implementation of the messaging layer from the remaining upper layers of the plugin. The group communication engine handles communications with the members of the replication group.

## 18.2 Getting Started

[18.2.1 Deploying Group Replication in Single-Primary Mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-deploying-in-single-primary-mode)

[18.2.2 Deploying Group Replication Locally](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-deploying-locally)

MySQL Group Replication is provided as a plugin to MySQL server, and each server in a group requires configuration and installation of the plugin. This section provides a detailed tutorial with the steps required to create a replication group with at least three members.

Tip

To deploy multiple instances of MySQL, you can use [InnoDB Cluster](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-cluster.html) which enables you to easily administer a group of MySQL server instances in [MySQL Shell](https://dev.mysql.com/doc/mysql-shell/8.0/en/). InnoDB Cluster wraps MySQL Group Replication in a programmatic environment that enables you easily deploy a cluster of MySQL instances to achieve high availability. In addition, InnoDB Cluster interfaces seamlessly with [MySQL Router](https://dev.mysql.com/doc/mysql-router/8.0/en/), which enables your applications to connect to the cluster without writing your own failover process. For similar use cases that do not require high availability, however, you can use [InnoDB ReplicaSet](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-replicaset.html). Installation instructions for MySQL Shell can be found [here](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-install.html).

### 18.2.1 Deploying Group Replication in Single-Primary Mode

[18.2.1.1 Deploying Instances for Group Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-getting-started-deploying-instances)

[18.2.1.2 Configuring an Instance for Group Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-configuring-instances)

[18.2.1.3 User Credentials For Distributed Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-user-credentials)

[18.2.1.4 Launching Group Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-launching)

[18.2.1.5 Bootstrapping the Group](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-bootstrap)

[18.2.1.6 Adding Instances to the Group](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-adding-instances)

Each of the MySQL server instances in a group can run on an independent physical host machine, which is the recommended way to deploy Group Replication. This section explains how to create a replication group with three MySQL Server instances, each running on a different host machine. See [Section 18.2.2, “Deploying Group Replication Locally”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-deploying-locally) for information about deploying multiple MySQL server instances running Group Replication on the same host machine, for example for testing purposes.

**Figure 18.7 Group Architecture**

This tutorial explains how to get and deploy MySQL Server with the Group Replication plugin, how to configure each server instance before creating a group, and how to use Performance Schema monitoring to verify that everything is working correctly.

#### 18.2.1.1 Deploying Instances for Group Replication

The first step is to deploy at least three instances of MySQL Server, this procedure demonstrates using multiple hosts for the instances, named s1, s2 and s3. It is assumed that MySQL Server was installed on each of the hosts, see [Chapter 2, *Installing and Upgrading MySQL*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html). Group Replication is a built-in MySQL plugin provided with MySQL Server 8.0, therefore no additional installation is required. For more background information on MySQL plugins, see [Section 5.6, “MySQL Server Plugins”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-plugins).

In this example, three instances are used for the group, which is the minimum number of instances to create a group. Adding more instances increases the fault tolerance of the group. For example if the group consists of three members, in event of failure of one instance the group can continue. But in the event of another failure the group can no longer continue processing write transactions. By adding more instances, the number of servers which can fail while the group continues to process transactions also increases. The maximum number of instances which can be used in a group is nine. For more information see [Section 18.1.4.2, “Failure Detection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-failure-detection).

#### 18.2.1.2 Configuring an Instance for Group Replication

This section explains the configuration settings required for MySQL Server instances that you want to use for Group Replication. For background information, see [Section 18.3, “Requirements and Limitations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-requirements-and-limitations).

[Storage Engines](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-storage-engines)

[Replication Framework](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-configure-replication-framework)

[Group Replication Settings](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-configure-plugin)

##### Storage Engines

For Group Replication, data must be stored in the InnoDB transactional storage engine (for details of why, see [Section 18.3.1, “Group Replication Requirements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-requirements)). The use of other storage engines, including the temporary [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) storage engine, might cause errors in Group Replication. Set the [**disabled\_storage\_engines**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_disabled_storage_engines) system variable as follows to prevent their use:

disabled\_storage\_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"

Note that with the [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) storage engine disabled, when you are upgrading a MySQL instance to a release where [**mysql\_upgrade**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql-upgrade) is still used (before MySQL 8.0.16), [**mysql\_upgrade**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql-upgrade) might fail with an error. To handle this, you can re-enable that storage engine while you run [**mysql\_upgrade**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql-upgrade), then disable it again when you restart the server. For more information, see [Section 4.4.5, “mysql\_upgrade — Check and Upgrade MySQL Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql-upgrade).

##### Replication Framework

The following settings configure replication according to the MySQL Group Replication requirements.

server\_id=1

gtid\_mode=ON

enforce\_gtid\_consistency=ON

These settings configure the server to use the unique identifier number 1, to enable [Section 17.1.3, “Replication with Global Transaction Identifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids), and to allow execution of only statements that can be safely logged using a GTID.

Up to and including MySQL 8.0.20, the following setting is also required:

binlog\_checksum=NONE

This setting disables checksums for events written to the binary log, which default to being enabled. From MySQL 8.0.21, Group Replication supports the presence of checksums in the binary log and can use them to verify the integrity of events on some channels, so you can use the default setting. For more details, see [Section 18.3.2, “Group Replication Limitations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-limitations).

If you are using a version of MySQL earlier than 8.0.3, where the defaults were improved for replication, you also need to add these lines to the member's option file. If you have any of these system variables in the option file in later versions, ensure that they are set as shown. For more details see [Section 18.3.1, “Group Replication Requirements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-requirements).

log\_bin=binlog

log\_slave\_updates=ON

binlog\_format=ROW

master\_info\_repository=TABLE

relay\_log\_info\_repository=TABLE

transaction\_write\_set\_extraction=XXHASH64

##### Group Replication Settings

At this point the option file ensures that the server is configured and is instructed to instantiate the replication infrastructure under a given configuration. The following section configures the Group Replication settings for the server.

plugin\_load\_add='group\_replication.so'

group\_replication\_group\_name="aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa"

group\_replication\_start\_on\_boot=off

group\_replication\_local\_address= "s1:33061"

group\_replication\_group\_seeds= "s1:33061,s2:33061,s3:33061"

group\_replication\_bootstrap\_group=off

plugin-load-add adds the Group Replication plugin to the list of plugins which the server loads at startup. This is preferable in a production deployment to installing the plugin manually.

Configuring [**group\_replication\_group\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_name) tells the plugin that the group that it is joining, or creating, is named "aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa".

The value of [**group\_replication\_group\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_name) must be a valid UUID. This UUID is used internally when setting GTIDs for Group Replication events in the binary log. You can use **SELECT UUID()** to generate a UUID.

Configuring the [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) variable to **off** instructs the plugin to not start operations automatically when the server starts. This is important when setting up Group Replication as it ensures you can configure the server before manually starting the plugin. Once the member is configured you can set [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) to **on** so that Group Replication starts automatically upon server boot.

Configuring [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) sets the network address and port which the member uses for internal communication with other members in the group. Group Replication uses this address for internal member-to-member connections involving remote instances of the group communication engine (XCom, a Paxos variant).

**Important**

The group replication local address must be different to the host name and port used for SQL client connections, which are defined by MySQL Server's [**hostname**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_hostname) and [**port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_port) system variables. It must not be used for client applications. It must be only be used for internal communication between the members of the group while running Group Replication.

The network address configured by [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) must be resolvable by all group members. For example, if each server instance is on a different machine with a fixed network address, you could use the IP address of the machine, such as 10.0.0.1. If you use a host name, you must use a fully qualified name, and ensure it is resolvable through DNS, correctly configured **/etc/hosts** files, or other name resolution processes. From MySQL 8.0.14, IPv6 addresses (or host names that resolve to them) can be used as well as IPv4 addresses. A group can contain a mix of members using IPv6 and members using IPv4. For more information on Group Replication support for IPv6 networks and on mixed IPv4 and IPv6 groups, see [Section 18.5.5, “Support For IPv6 And For Mixed IPv6 And IPv4 Groups”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-ipv6).

The recommended port for [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) is 33061. [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) is used by Group Replication as the unique identifier for a group member within the replication group. You can use the same port for all members of a replication group as long as the host names or IP addresses are all different, as demonstrated in this tutorial. Alternatively you can use the same host name or IP address for all members as long as the ports are all different, for example as shown in [Section 18.2.2, “Deploying Group Replication Locally”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-deploying-locally).

The connection that an existing member offers to a joining member for Group Replication's distributed recovery process is not the network address configured by [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address). Up to MySQL 8.0.20, group members offer their standard SQL client connection to joining members for distributed recovery, as specified by MySQL Server's [**hostname**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_hostname) and [**port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_port) system variables. From MySQL 8.0.21, group members may advertise an alternative list of distributed recovery endpoints as dedicated client connections for joining members. For more details, see [Section 18.5.3.1, “Connections for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-connections).

**Important**

Distributed recovery can fail if a joining member cannot correctly identify the other members using the host name as defined by MySQL Server's [**hostname**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_hostname) system variable. It is recommended that operating systems running MySQL have a properly configured unique host name, either using DNS or local settings. The host name that the server is using for SQL client connections can be verified in the **Member\_host** column of the Performance Schema table [**replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table). If multiple group members externalize a default host name set by the operating system, there is a chance of the joining member not resolving it to the correct member address and not being able to connect for distributed recovery. In this situation you can use MySQL Server's [**report\_host**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_host) system variable to configure a unique host name to be externalized by each of the servers.

Configuring [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) sets the hostname and port of the group members which are used by the new member to establish its connection to the group. These members are called the seed members. Once the connection is established, the group membership information is listed in the Performance Schema table [**replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table). Usually the [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) list contains the **hostname:port** of each of the group member's [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address), but this is not obligatory and a subset of the group members can be chosen as seeds.

**Important**

The **hostname:port** listed in [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) is the seed member's internal network address, configured by [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) and not the **hostname:port** used for SQL client connections, which is shown for example in the Performance Schema table [**replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table).

The server that starts the group does not make use of this option, since it is the initial server and as such, it is in charge of bootstrapping the group. In other words, any existing data which is on the server bootstrapping the group is what is used as the data for the next joining member. The second server joining asks the one and only member in the group to join, any missing data on the second server is replicated from the donor data on the bootstrapping member, and then the group expands. The third server joining can ask any of these two to join, data is synchronized to the new member, and then the group expands again. Subsequent servers repeat this procedure when joining.

**Warning**

When joining multiple servers at the same time, make sure that they point to seed members that are already in the group. Do not use members that are also joining the group as seeds, because they might not yet be in the group when contacted.

It is good practice to start the bootstrap member first, and let it create the group. Then make it the seed member for the rest of the members that are joining. This ensures that there is a group formed when joining the rest of the members.

Creating a group and joining multiple members at the same time is not supported. It might work, but chances are that the operations race and then the act of joining the group ends up in an error or a time out.

A joining member must communicate with a seed member using the same protocol (IPv4 or IPv6) that the seed member advertises in the [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) option. For the purpose of IP address permissions for Group Replication, the allowlist on the seed member must include an IP address for the joining member for the protocol offered by the seed member, or a host name that resolves to an address for that protocol. This address or host name must be set up and permitted in addition to the joining member's [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) if the protocol for that address does not match the seed member's advertised protocol. If a joining member does not have a permitted address for the appropriate protocol, its connection attempt is refused. For more information, see [Section 18.6.1, “Group Replication IP Address Permissions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-ip-address-permissions).

Configuring [**group\_replication\_bootstrap\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group) instructs the plugin whether to bootstrap the group or not. In this case, even though s1 is the first member of the group we set this variable to off in the option file. Instead we configure [**group\_replication\_bootstrap\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group) when the instance is running, to ensure that only one member actually bootstraps the group.

**Important**

The [**group\_replication\_bootstrap\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group) variable must only be enabled on one server instance belonging to a group at any time, usually the first time you bootstrap the group (or in case the entire group is brought down and back up again). If you bootstrap the group multiple times, for example when multiple server instances have this option set, then they could create an artificial split brain scenario, in which two distinct groups with the same name exist. Always set [**group\_replication\_bootstrap\_group=off**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group) after the first server instance comes online.

Configuration for all servers in the group is quite similar. You need to change the specifics about each server (for example [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id), [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir), [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address)). This is illustrated later in this tutorial.

#### 18.2.1.3 User Credentials For Distributed Recovery

Group Replication uses a distributed recovery process to synchronize group members when joining them to the group. Distributed recovery involves transferring transactions from a donor's binary log to a joining member using a replication channel named **group\_replication\_recovery**. You must therefore set up a replication user with the correct permissions so that Group Replication can establish direct member-to-member replication channels. If group members have been set up to support the use of a remote cloning operation as part of distributed recovery, which is available from MySQL 8.0.17, this replication user is also used as the clone user on the donor, and requires the correct permissions for this role too. For a complete description of distributed recovery, see [Section 18.5.3, “Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery).

The same replication user must be used for distributed recovery on every group member. The process of creating the replication user for distributed recovery can be captured in the binary log, and then you can rely on distributed recovery to replicate the statements used to create the user. Alternatively, you can disable binary logging before creating the replication user, and then create the user manually on each member, for example if you want to avoid the changes being propagated to other server instances. If you do this, ensure you re-enable binary logging once you have configured the user.

**Important**

If distributed recovery connections for your group use SSL, the replication user must be created on each server before the joining member connects to the donor. For instructions to set up SSL for distributed recovery connections and create a replication user that requires SSL, see [Section 18.6.3, “Securing Distributed Recovery Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-securing)

**Important**

By default, users created in MySQL 8 use [Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#caching-sha2-pluggable-authentication). If the replication user for distributed recovery uses the caching SHA-2 authentication plugin, and you are not using SSL for distributed recovery connections, RSA key-pairs are used for password exchange. You can either copy the public key of the replication user to the joining member, or configure the donors to provide the public key when requested. For instructions to do this, see [Section 18.6.3.1, “Secure User Credentials for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-user).

To create the replication user for distributed recovery, follow these steps:

Start the MySQL server instance, then connect a client to it.

If you want to disable binary logging in order to create the replication user separately on each instance, do so by issuing the following statement:

mysql> **SET SQL\_LOG\_BIN=0;**

Create a MySQL user with the [**REPLICATION SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave) privilege to use for distributed recovery, and if the server is set up to support cloning, the [**BACKUP\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_backup-admin) privilege to use as the donor in a cloning operation. In this example the user ***rpl\_user*** with the password ***password*** is shown. When configuring your servers use a suitable user name and password:

mysql> **CREATE USER *rpl\_user*@'%' IDENTIFIED BY '*password*';**

mysql> **GRANT REPLICATION SLAVE ON \*.\* TO *rpl\_user*@'%';**

mysql> **GRANT BACKUP\_ADMIN ON \*.\* TO *rpl\_user*@'%';**

mysql> **FLUSH PRIVILEGES;**

If you disabled binary logging, enable it again as soon as you have created the user, by issuing the following statement:

mysql> **SET SQL\_LOG\_BIN=1;**

When you have created the replication user, you must supply the user credentials to the server for use with distributed recovery. You can do this by setting the user credentials as the credentials for the **group\_replication\_recovery** channel, using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). Alternatively, from MySQL 8.0.21, you can specify the user credentials for distributed recovery on the [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement.

User credentials set using [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) are stored in plain text in the replication metadata repositories on the server. They are applied whenever Group Replication is started, including automatic starts if the [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) system variable is set to **ON**.

User credentials specified on [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) are saved in memory only, and are removed by a [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) statement or server shutdown. You must issue a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement to provide the credentials again, so you cannot start Group Replication automatically with these credentials. This method of specifying the user credentials helps to secure the Group Replication servers against unauthorized access.

For more information on the security implications of each method of providing the user credentials, see [Section 18.6.3.1.3, “Providing Replication User Credentials Securely”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-user-provide). If you choose to provide the user credentials using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, issue the following statement on the server instance now, replacing ***rpl\_user*** and ***password*** with the values used when creating the user:

mysql> **CHANGE MASTER TO MASTER\_USER='*rpl\_user*', MASTER\_PASSWORD='*password*' \\**

**FOR CHANNEL 'group\_replication\_recovery';**

Or from MySQL 8.0.23:

mysql> **CHANGE REPLICATION SOURCE TO SOURCE\_USER='*rpl\_user*', SOURCE\_PASSWORD='*password*' \\**

**FOR CHANNEL 'group\_replication\_recovery';**

#### 18.2.1.4 Launching Group Replication

Once server s1 has been configured and started, install the Group Replication plugin. If you used **plugin\_load\_add='group\_replication.so'** in the option file then the Group Replication plugin is installed and you can proceed to the next step. In the event that you decide to install the plugin manually, connect to the server and issue the following:

**INSTALL PLUGIN group\_replication SONAME 'group\_replication.so';**

**Important**

The **mysql.session** user must exist before you can load Group Replication. **mysql.session** was added in MySQL version 8.0.2. If your data dictionary was initialized using an earlier version you must perform the MySQL upgrade procedure (see [Section 2.11, “Upgrading MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading)). If the upgrade is not run, Group Replication fails to start with the error message There was an error when trying to access the server with user: mysql.session@localhost. Make sure the user is present in the server and that mysql\_upgrade was ran after a server update..

To check that the plugin was installed successfully, issue **SHOW PLUGINS;** and check the output. It should show something like this:

mysql> **SHOW PLUGINS;**

+----------------------------+----------+--------------------+----------------------+-------------+

| Name | Status | Type | Library | License |

+----------------------------+----------+--------------------+----------------------+-------------+

| binlog | ACTIVE | STORAGE ENGINE | NULL | PROPRIETARY |

(...)

| group\_replication | ACTIVE | GROUP REPLICATION | group\_replication.so | PROPRIETARY |

+----------------------------+----------+--------------------+----------------------+-------------+

#### 18.2.1.5 Bootstrapping the Group

The process of starting a group for the first time is called bootstrapping. You use the [**group\_replication\_bootstrap\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group) system variable to bootstrap a group. The bootstrap should only be done by a single server, the one that starts the group and only once. This is why the value of the [**group\_replication\_bootstrap\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group) option was not stored in the instance's option file. If it is saved in the option file, upon restart the server automatically bootstraps a second group with the same name. This would result in two distinct groups with the same name. The same reasoning applies to stopping and restarting the plugin with this option set to **ON**. Therefore to safely bootstrap the group, connect to s1 and issue the following statements:

mysql> **SET GLOBAL group\_replication\_bootstrap\_group=ON;**

mysql> **START GROUP\_REPLICATION;**

mysql> **SET GLOBAL group\_replication\_bootstrap\_group=OFF;**

Or if you are providing user credentials for distributed recovery on the [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement (which you can from MySQL 8.0.21), issue the following statements:

mysql> **SET GLOBAL group\_replication\_bootstrap\_group=ON;**

mysql> **START GROUP\_REPLICATION USER='*rpl\_user*', PASSWORD='*password*';**

mysql> **SET GLOBAL group\_replication\_bootstrap\_group=OFF;**

Once the [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement returns, the group has been started. You can check that the group is now created and that there is one member in it:

mysql> **SELECT \* FROM performance\_schema.replication\_group\_members;**

+---------------------------+--------------------------------------+-------------+-------------+---------------+

| CHANNEL\_NAME | MEMBER\_ID | MEMBER\_HOST | MEMBER\_PORT | MEMBER\_STATE |

+---------------------------+--------------------------------------+-------------+-------------+---------------+

| group\_replication\_applier | ce9be252-2b71-11e6-b8f4-00212844f856 | s1 | 3306 | ONLINE |

+---------------------------+--------------------------------------+-------------+-------------+---------------+

The information in this table confirms that there is a member in the group with the unique identifier **ce9be252-2b71-11e6-b8f4-00212844f856**, that it is **ONLINE** and is at **s1** listening for client connections on port **3306**.

For the purpose of demonstrating that the server is indeed in a group and that it is able to handle load, create a table and add some content to it.

mysql> **CREATE DATABASE test;**

mysql> **USE test;**

mysql> **CREATE TABLE t1 (c1 INT PRIMARY KEY, c2 TEXT NOT NULL);**

mysql> **INSERT INTO t1 VALUES (1, 'Luis');**

Check the content of table **t1** and the binary log.

mysql> **SELECT \* FROM t1;**

+----+------+

| c1 | c2 |

+----+------+

| 1 | Luis |

+----+------+

mysql> **SHOW BINLOG EVENTS;**

+---------------+-----+----------------+-----------+-------------+--------------------------------------------------------------------+

| Log\_name | Pos | Event\_type | Server\_id | End\_log\_pos | Info |

+---------------+-----+----------------+-----------+-------------+--------------------------------------------------------------------+

| binlog.000001 | 4 | Format\_desc | 1 | 123 | Server ver: 8.0.26-log, Binlog ver: 4 |

| binlog.000001 | 123 | Previous\_gtids | 1 | 150 | |

| binlog.000001 | 150 | Gtid | 1 | 211 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:1' |

| binlog.000001 | 211 | Query | 1 | 270 | BEGIN |

| binlog.000001 | 270 | View\_change | 1 | 369 | view\_id=14724817264259180:1 |

| binlog.000001 | 369 | Query | 1 | 434 | COMMIT |

| binlog.000001 | 434 | Gtid | 1 | 495 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:2' |

| binlog.000001 | 495 | Query | 1 | 585 | CREATE DATABASE test |

| binlog.000001 | 585 | Gtid | 1 | 646 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:3' |

| binlog.000001 | 646 | Query | 1 | 770 | use `test`; CREATE TABLE t1 (c1 INT PRIMARY KEY, c2 TEXT NOT NULL) |

| binlog.000001 | 770 | Gtid | 1 | 831 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:4' |

| binlog.000001 | 831 | Query | 1 | 899 | BEGIN |

| binlog.000001 | 899 | Table\_map | 1 | 942 | table\_id: 108 (test.t1) |

| binlog.000001 | 942 | Write\_rows | 1 | 984 | table\_id: 108 flags: STMT\_END\_F |

| binlog.000001 | 984 | Xid | 1 | 1011 | COMMIT /\* xid=38 \*/ |

+---------------+-----+----------------+-----------+-------------+--------------------------------------------------------------------+

As seen above, the database and the table objects were created and their corresponding DDL statements were written to the binary log. Also, the data was inserted into the table and written to the binary log, so it can be used for distributed recovery by state transfer from a donor's binary log.

#### 18.2.1.6 Adding Instances to the Group

At this point, the group has one member in it, server s1, which has some data in it. It is now time to expand the group by adding the other two servers configured previously.

##### 18.2.1.6.1 Adding a Second Instance

In order to add a second instance, server s2, first create the configuration file for it. The configuration is similar to the one used for server s1, except for things such as the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id). These different lines are highlighted in the listing below.

[mysqld]

#

# Disable other storage engines

#

disabled\_storage\_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"

#

# Replication configuration parameters

#

server\_id=2

gtid\_mode=ON

enforce\_gtid\_consistency=ON

binlog\_checksum=NONE # Not needed from 8.0.21

#

# Group Replication configuration

#

plugin\_load\_add='group\_replication.so'

group\_replication\_group\_name="aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa"

group\_replication\_start\_on\_boot=off

group\_replication\_local\_address= "s2:33061"

group\_replication\_group\_seeds= "s1:33061,s2:33061,s3:33061"

group\_replication\_bootstrap\_group= off

Similar to the procedure for server s1, with the option file in place you launch the server. Then configure the distributed recovery credentials as follows. The commands are the same as used when setting up server s1 as the user is shared within the group. This member needs to have the same replication user configured in [Section 18.2.1.3, “User Credentials For Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-user-credentials). If you are relying on distributed recovery to configure the user on all members, when s2 connects to the seed s1 the replication user is replicated or cloned to s1. If you did not have binary logging enabled when you configured the user credentials on s1, and a remote cloning operation is not used for state transfer, you must create the replication user on s2. In this case, connect to s2 and issue:

**SET SQL\_LOG\_BIN=0;**

**CREATE USER *rpl\_user*@'%' IDENTIFIED BY '*password*';**

**GRANT REPLICATION SLAVE ON \*.\* TO *rpl\_user*@'%';**

**GRANT BACKUP\_ADMIN ON \*.\* TO *rpl\_user*@'%';**

**FLUSH PRIVILEGES;**

**SET SQL\_LOG\_BIN=1;**

If you are providing user credentials using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, issue the following statement after that:

**CHANGE MASTER TO MASTER\_USER='*rpl\_user*', MASTER\_PASSWORD='*password*' \\**

**FOR CHANNEL 'group\_replication\_recovery';**

Or from MySQL 8.0.23:

**CHANGE REPLICATION SOURCE TO SOURCE\_USER='*rpl\_user*', SOURCE\_PASSWORD='*password*' \\**

**FOR CHANNEL 'group\_replication\_recovery';**

Tip

If you are using the caching SHA-2 authentication plugin, the default in MySQL 8, see [Section 18.6.3.1.1, “Replication User With The Caching SHA-2 Authentication Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-caching-sha2-user-credentials).

If necessary, install the Group Replication plugin, see [Section 18.2.1.4, “Launching Group Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-launching).

Start Group Replication and s2 starts the process of joining the group.

mysql> **START GROUP\_REPLICATION;**

Or if you are providing user credentials for distributed recovery on the [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement (which you can from MySQL 8.0.21):

mysql> **START GROUP\_REPLICATION USER='*rpl\_user*', PASSWORD='*password*';**

Unlike the previous steps that were the same as those executed on s1, here there is a difference in that you do not need to bootstrap the group because the group already exists. In other words on s2 [**group\_replication\_bootstrap\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group) is set to **OFF**, and you do not issue **SET GLOBAL group\_replication\_bootstrap\_group=ON;** before starting Group Replication, because the group has already been created and bootstrapped by server s1. At this point server s2 only needs to be added to the already existing group.

Tip

When Group Replication starts successfully and the server joins the group it checks the [**super\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) variable. By setting [**super\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) to ON in the member's configuration file, you can ensure that servers which fail when starting Group Replication for any reason do not accept transactions. If the server should join the group as a read-write instance, for example as the primary in a single-primary group or as a member of a multi-primary group, when the [**super\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) variable is set to ON then it is set to OFF upon joining the group.

Checking the [**performance\_schema.replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table) table again shows that there are now two **ONLINE** servers in the group.

mysql> **SELECT \* FROM performance\_schema.replication\_group\_members;**

+---------------------------+--------------------------------------+-------------+-------------+---------------+

| CHANNEL\_NAME | MEMBER\_ID | MEMBER\_HOST | MEMBER\_PORT | MEMBER\_STATE |

+---------------------------+--------------------------------------+-------------+-------------+---------------+

| group\_replication\_applier | 395409e1-6dfa-11e6-970b-00212844f856 | s1 | 3306 | ONLINE |

| group\_replication\_applier | ac39f1e6-6dfa-11e6-a69d-00212844f856 | s2 | 3306 | ONLINE |

+---------------------------+--------------------------------------+-------------+-------------+---------------+

When s2 attempted to join the group, [Section 18.5.3, “Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery) ensured that s2 applied the same transactions which s1 had applied. Once this process completed, s2 could join the group as a member, and at this point it is marked as **ONLINE**. In other words it must have already caught up with server s1 automatically. Once s2 is **ONLINE**, it then begins to process transactions with the group. Verify that s2 has indeed synchronized with server s1 as follows.

mysql> **SHOW DATABASES LIKE 'test';**

+-----------------+

| Database (test) |

+-----------------+

| test |

+-----------------+

mysql> **SELECT \* FROM test.t1;**

+----+------+

| c1 | c2 |

+----+------+

| 1 | Luis |

+----+------+

mysql> **SHOW BINLOG EVENTS;**

+---------------+------+----------------+-----------+-------------+--------------------------------------------------------------------+

| Log\_name | Pos | Event\_type | Server\_id | End\_log\_pos | Info |

+---------------+------+----------------+-----------+-------------+--------------------------------------------------------------------+

| binlog.000001 | 4 | Format\_desc | 2 | 123 | Server ver: 8.0.26-log, Binlog ver: 4 |

| binlog.000001 | 123 | Previous\_gtids | 2 | 150 | |

| binlog.000001 | 150 | Gtid | 1 | 211 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:1' |

| binlog.000001 | 211 | Query | 1 | 270 | BEGIN |

| binlog.000001 | 270 | View\_change | 1 | 369 | view\_id=14724832985483517:1 |

| binlog.000001 | 369 | Query | 1 | 434 | COMMIT |

| binlog.000001 | 434 | Gtid | 1 | 495 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:2' |

| binlog.000001 | 495 | Query | 1 | 585 | CREATE DATABASE test |

| binlog.000001 | 585 | Gtid | 1 | 646 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:3' |

| binlog.000001 | 646 | Query | 1 | 770 | use `test`; CREATE TABLE t1 (c1 INT PRIMARY KEY, c2 TEXT NOT NULL) |

| binlog.000001 | 770 | Gtid | 1 | 831 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:4' |

| binlog.000001 | 831 | Query | 1 | 890 | BEGIN |

| binlog.000001 | 890 | Table\_map | 1 | 933 | table\_id: 108 (test.t1) |

| binlog.000001 | 933 | Write\_rows | 1 | 975 | table\_id: 108 flags: STMT\_END\_F |

| binlog.000001 | 975 | Xid | 1 | 1002 | COMMIT /\* xid=30 \*/ |

| binlog.000001 | 1002 | Gtid | 1 | 1063 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:5' |

| binlog.000001 | 1063 | Query | 1 | 1122 | BEGIN |

| binlog.000001 | 1122 | View\_change | 1 | 1261 | view\_id=14724832985483517:2 |

| binlog.000001 | 1261 | Query | 1 | 1326 | COMMIT |

+---------------+------+----------------+-----------+-------------+--------------------------------------------------------------------+

As seen above, the second server has been added to the group and it has replicated the changes from server s1 automatically. In other words, the transactions applied on s1 up to the point in time that s2 joined the group have been replicated to s2.

##### 18.2.1.6.2 Adding Additional Instances

Adding additional instances to the group is essentially the same sequence of steps as adding the second server, except that the configuration has to be changed as it had to be for server s2. To summarise the required commands:

Create the configuration file.

[mysqld]

#

# Disable other storage engines

#

disabled\_storage\_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"

#

# Replication configuration parameters

#

server\_id=3

gtid\_mode=ON

enforce\_gtid\_consistency=ON

binlog\_checksum=NONE # Not needed from 8.0.21

#

# Group Replication configuration

#

plugin\_load\_add='group\_replication.so'

group\_replication\_group\_name="aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa"

group\_replication\_start\_on\_boot=off

group\_replication\_local\_address= "s3:33061"

group\_replication\_group\_seeds= "s1:33061,s2:33061,s3:33061"

group\_replication\_bootstrap\_group= off

Start the server and connect to it. Create the replication user for distributed recovery.

**SET SQL\_LOG\_BIN=0;**

**CREATE USER *rpl\_user*@'%' IDENTIFIED BY '*password*';**

**GRANT REPLICATION SLAVE ON \*.\* TO *rpl\_user*@'%';**

**GRANT BACKUP\_ADMIN ON \*.\* TO *rpl\_user*@'%';**

**FLUSH PRIVILEGES;**

**SET SQL\_LOG\_BIN=1;**

If you are providing user credentials using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, issue the following statement after that:

**CHANGE MASTER TO MASTER\_USER='*rpl\_user*', MASTER\_PASSWORD='*password*' \\**

**FOR CHANNEL 'group\_replication\_recovery';**

Or from MySQL 8.0.23:

**CHANGE REPLICATION SOURCE TO SOURCE\_USER='*rpl\_user*', SOURCE\_PASSWORD='*password*' \\**

**FOR CHANNEL 'group\_replication\_recovery';**

Install the Group Replication plugin if necessary.

**INSTALL PLUGIN group\_replication SONAME 'group\_replication.so';**

Start Group Replication.

mysql> **START GROUP\_REPLICATION;**

Or if you are providing user credentials for distributed recovery on the [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement (which you can from MySQL 8.0.21):

mysql> **START GROUP\_REPLICATION USER='*rpl\_user*', PASSWORD='*password*';**

At this point server s3 is booted and running, has joined the group and caught up with the other servers in the group. Consulting the [**performance\_schema.replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table) table again confirms this is the case.

mysql> SELECT \* FROM performance\_schema.replication\_group\_members;

+---------------------------+--------------------------------------+-------------+-------------+---------------+

| CHANNEL\_NAME | MEMBER\_ID | MEMBER\_HOST | MEMBER\_PORT | MEMBER\_STATE |

+---------------------------+--------------------------------------+-------------+-------------+---------------+

| group\_replication\_applier | 395409e1-6dfa-11e6-970b-00212844f856 | s1 | 3306 | ONLINE |

| group\_replication\_applier | 7eb217ff-6df3-11e6-966c-00212844f856 | s3 | 3306 | ONLINE |

| group\_replication\_applier | ac39f1e6-6dfa-11e6-a69d-00212844f856 | s2 | 3306 | ONLINE |

+---------------------------+--------------------------------------+-------------+-------------+---------------+

Issuing this same query on server s2 or server s1 yields the same result. Also, you can verify that server s3 has caught up:

mysql> **SHOW DATABASES LIKE 'test';**

+-----------------+

| Database (test) |

+-----------------+

| test |

+-----------------+

mysql> **SELECT \* FROM test.t1;**

+----+------+

| c1 | c2 |

+----+------+

| 1 | Luis |

+----+------+

mysql> **SHOW BINLOG EVENTS;**

+---------------+------+----------------+-----------+-------------+--------------------------------------------------------------------+

| Log\_name | Pos | Event\_type | Server\_id | End\_log\_pos | Info |

+---------------+------+----------------+-----------+-------------+--------------------------------------------------------------------+

| binlog.000001 | 4 | Format\_desc | 3 | 123 | Server ver: 8.0.26-log, Binlog ver: 4 |

| binlog.000001 | 123 | Previous\_gtids | 3 | 150 | |

| binlog.000001 | 150 | Gtid | 1 | 211 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:1' |

| binlog.000001 | 211 | Query | 1 | 270 | BEGIN |

| binlog.000001 | 270 | View\_change | 1 | 369 | view\_id=14724832985483517:1 |

| binlog.000001 | 369 | Query | 1 | 434 | COMMIT |

| binlog.000001 | 434 | Gtid | 1 | 495 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:2' |

| binlog.000001 | 495 | Query | 1 | 585 | CREATE DATABASE test |

| binlog.000001 | 585 | Gtid | 1 | 646 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:3' |

| binlog.000001 | 646 | Query | 1 | 770 | use `test`; CREATE TABLE t1 (c1 INT PRIMARY KEY, c2 TEXT NOT NULL) |

| binlog.000001 | 770 | Gtid | 1 | 831 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:4' |

| binlog.000001 | 831 | Query | 1 | 890 | BEGIN |

| binlog.000001 | 890 | Table\_map | 1 | 933 | table\_id: 108 (test.t1) |

| binlog.000001 | 933 | Write\_rows | 1 | 975 | table\_id: 108 flags: STMT\_END\_F |

| binlog.000001 | 975 | Xid | 1 | 1002 | COMMIT /\* xid=29 \*/ |

| binlog.000001 | 1002 | Gtid | 1 | 1063 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:5' |

| binlog.000001 | 1063 | Query | 1 | 1122 | BEGIN |

| binlog.000001 | 1122 | View\_change | 1 | 1261 | view\_id=14724832985483517:2 |

| binlog.000001 | 1261 | Query | 1 | 1326 | COMMIT |

| binlog.000001 | 1326 | Gtid | 1 | 1387 | SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:6' |

| binlog.000001 | 1387 | Query | 1 | 1446 | BEGIN |

| binlog.000001 | 1446 | View\_change | 1 | 1585 | view\_id=14724832985483517:3 |

| binlog.000001 | 1585 | Query | 1 | 1650 | COMMIT |

+---------------+------+----------------+-----------+-------------+--------------------------------------------------------------------+

### 18.2.2 Deploying Group Replication Locally

The most common way to deploy Group Replication is using multiple server instances, to provide high availability. It is also possible to deploy Group Replication locally, for example for testing purposes. This section explains how you can deploy Group Replication locally.

**Important**

Group Replication is usually deployed on multiple hosts because this ensures that high-availability is provided. The instructions in this section are not suitable for production deployments because all MySQL server instances are running on the same single host. In the event of failure of this host, the whole group fails. Therefore this information should be used for testing purposes and it should not be used in a production environments.

This section explains how to create a replication group with three MySQL Server instances on one physical machine. This means that three data directories are needed, one per server instance, and that you need to configure each instance independently. This - procedure assumes that MySQL Server was downloaded and unpacked - into the directory named **mysql-8.0**. Each MySQL server instance requires a specific data directory. Create a directory named **data**, then in that directory create a subdirectory for each server instance, for example s1, s2 and s3, and initialize each one.

**mysql-8.0/bin/mysqld --initialize-insecure --basedir=$PWD/mysql-8.0 --datadir=$PWD/data/s1**

**mysql-8.0/bin/mysqld --initialize-insecure --basedir=$PWD/mysql-8.0 --datadir=$PWD/data/s2**

**mysql-8.0/bin/mysqld --initialize-insecure --basedir=$PWD/mysql-8.0 --datadir=$PWD/data/s3**

Inside **data/s1**, **data/s2**, **data/s3** is an initialized data directory, containing the mysql system database and related tables and much more. To learn more about the initialization procedure, see [Section 2.10.1, “Initializing the Data Directory”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#data-directory-initialization).

**Warning**

Do not use **-initialize-insecure** in production environments, it is only used here to simplify the tutorial. For more information on security settings, see [Section 18.6, “Group Replication Security”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-security).

#### Configuration of Local Group Replication Members

When you are following [Section 18.2.1.2, “Configuring an Instance for Group Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-configuring-instances), you need to add configuration for the data directories added in the previous section. For example:

[mysqld]

# server configuration

datadir=<full\_path\_to\_data>/data/s1

basedir=<full\_path\_to\_bin>/mysql-8.0/

port=24801

socket=<full\_path\_to\_sock\_dir>/s1.sock

These settings configure MySQL server to use the data directory created earlier and which port the server should open and start listening for incoming connections.

**Note**

The non-default port of 24801 is used because in this tutorial the three server instances use the same hostname. In a setup with three different machines this would not be required.

Group Replication requires a network connection between the members, which means that each member must be able to resolve the network address of all of the other members. For example in this tutorial all three instances run on one machine, so to ensure that the members can contact each other you could add a line to the option file such as [**report\_host=127.0.0.1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_host).

Then each member needs to be able to connect to the other members on their [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address). For example in the option file of member s1 add:

group\_replication\_local\_address= "127.0.0.1:24901"

group\_replication\_group\_seeds= "127.0.0.1:24901,127.0.0.1:24902,127.0.0.1:24903"

This configures s1 to use port 24901 for internal group communication with seed members. For each server instance you want to add to the group, make these changes in the option file of the member. For each member you must ensure a unique address is specified, so use a unique port per instance for [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address). Usually you want all members to be able to serve as seeds for members that are joining the group and have not got the transactions processed by the group. In this case, add all of the ports to [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) as shown above.

The remaining steps of [Section 18.2.1, “Deploying Group Replication in Single-Primary Mode”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-deploying-in-single-primary-mode) apply equally to a group which you have deployed locally in this way.

## 18.3 Requirements and Limitations

[18.3.1 Group Replication Requirements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-requirements)

[18.3.2 Group Replication Limitations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-limitations)

This section lists and explains the requirements and limitations of Group Replication.

### 18.3.1 Group Replication Requirements

Server instances that you want to use for Group Replication must satisfy the following requirements.

#### Infrastructure

**InnoDB Storage Engine.** Data must be stored in the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) transactional storage engine. Transactions are executed optimistically and then, at commit time, are checked for conflicts. If there are conflicts, in order to maintain consistency across the group, some transactions are rolled back. This means that a transactional storage engine is required. Moreover, [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) provides some additional functionality that enables better management and handling of conflicts when operating together with Group Replication. The use of other storage engines, including the temporary [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) storage engine, might cause errors in Group Replication. You can prevent the use of other storage engines by setting the [**disabled\_storage\_engines**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_disabled_storage_engines) system variable on group members, for example:

disabled\_storage\_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"

**Primary Keys.** Every table that is to be replicated by the group must have a defined primary key, or primary key equivalent where the equivalent is a non-null unique key. Such keys are required as a unique identifier for every row within a table, enabling the system to determine which transactions conflict by identifying exactly which rows each transaction has modified. Group Replication has its own built-in set of checks for primary keys or primary key equivalents, and does not use the checks carried out by the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable. You may set **sql\_require\_primary\_key=ON** for a server instance where Group Replication is running, and you may set the **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement to **ON** for a Group Replication channel. However, be aware that you might find some transactions that are permitted under Group Replication's built-in checks are not permitted under the checks carried out when you set **sql\_require\_primary\_key=ON** or **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK=ON**.

**Network Performance.** MySQL Group Replication is designed to be deployed in a cluster environment where server instances are very close to each other. The performance and stability of a group can be impacted by both network latency and network bandwidth. Bi-directional communication must be maintained at all times between all group members. If either inbound or outbound communication is blocked for a server instance (for example, by a firewall, or by connectivity issues), the member cannot function in the group, and the group members (including the member with issues) might not be able to report the correct member status for the affected server instance.

From MySQL 8.0.14, you can use an IPv4 or IPv6 network infrastructure, or a mix of the two, for TCP communication between remote Group Replication servers. There is also nothing preventing Group Replication from operating over a virtual private network (VPN).

Also from MySQL 8.0.14, where Group Replication server instances are co-located and share a local group communication engine (XCom) instance, a dedicated input channel with lower overhead is used for communication where possible instead of the TCP socket. For certain Group Replication tasks that require communication between remote XCom instances, such as joining a group, the TCP network is still used, so network performance influences the group's performance.

#### Server Instance Configuration

The following options must be configured as shown on server instances that are members of a group.

**Unique Server Identifier.** Use the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) system variable to configure the server with a unique server ID, as required for all servers in replication topologies. The server ID must be a positive integer between 1 and (232)−1, and it must be different from every other server ID in use by any other server in the replication topology.

**Binary Log Active.** Set [**--log-bin[=log\_file\_name]**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin). MySQL Group Replication replicates binary log contents, therefore the binary log needs to be on for it to operate. This option is enabled by default. See [Section 5.4.4, “The Binary Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log).

**Replica Updates Logged.** Set [**--log-slave-updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates). This option is enabled by default. Group members need to log transactions that are received from their donors at joining time and applied through the replication applier, and to log all transactions that they receive and apply from the group. This enables Group Replication to carry out distributed recovery by state transfer from an existing group member's binary log.

**Binary Log Row Format.** Set [**--binlog-format=row**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format). Group Replication relies on row-based replication format to propagate changes consistently among the servers in the group. It relies on row-based infrastructure to be able to extract the necessary information to detect conflicts among transactions that execute concurrently in different servers in the group. From MySQL 8.0.19, the **REQUIRE\_ROW\_FORMAT** setting is automatically added to Group Replication's channels to enforce the use of row-based replication when the transactions are applied. See [Section 17.2.1, “Replication Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-formats) and [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks).

**Binary Log Checksums Off (to MySQL 8.0.20).** Up to and including MySQL 8.0.20, set [**--binlog-checksum=NONE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_checksum). In these releases, Group Replication cannot make use of checksums and does not support their presence in the binary log. From MySQL 8.0.21, Group Replication supports checksums, so group members may use the default setting.

**Global Transaction Identifiers On.** Set [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) and [**enforce\_gtid\_consistency=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency). Group Replication uses global transaction identifiers to track exactly which transactions have been committed on every server instance and thus be able to infer which servers have executed transactions that could conflict with already committed transactions elsewhere. In other words, explicit transaction identifiers are a fundamental part of the framework to be able to determine which transactions may conflict. See [Section 17.1.3, “Replication with Global Transaction Identifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids).

**Replication Information Repositories.** Set [**master\_info\_repository=TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_master_info_repository) and [**relay\_log\_info\_repository=TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_repository). In MySQL 8.0, this setting is the default, and the **FILE** setting is deprecated. From MySQL 8.0.23, the use of these system variables is deprecated, so omit the system variables and just allow the default. The replication applier needs to have the replication metadata written to the **mysql.slave\_master\_info** and **mysql.slave\_relay\_log\_info** system tables to ensure the Group Replication plugin has consistent recoverability and transactional management of the replication metadata. See [Section 17.2.4.2, “Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs-status).

**Transaction Write Set Extraction.** Set [**--transaction-write-set-extraction=XXHASH64**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_transaction_write_set_extraction) so that while collecting rows to log them to the binary log, the server collects the write set as well. The write set is based on the primary keys of each row and is a simplified and compact view of a tag that uniquely identifies the row that was changed. This tag is then used for detecting conflicts. This option is enabled by default.

**Binary Log Dependency Tracking.** Setting [**binlog\_transaction\_dependency\_tracking=WRITESET\_SESSION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking) can improve performance for a group member, depending on the group's workload. Group Replication carries out its own parallelization after certification when applying transactions from the relay log, independently of the value set for [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking). However, the value of [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking) does affect how transactions are written to the binary logs on Group Replication members. The dependency information in those logs is used to assist the process of state transfer from a donor's binary log for distributed recovery, which takes place whenever a member joins or rejoins the group.

**Default Table Encryption.** Set [**--default-table-encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) to the same value on all group members. Default schema and tablespace encryption can be either enabled (**ON**) or disabled (**OFF**, the default) as long as the setting is the same on all members.

**Lower Case Table Names.** Set [**--lower-case-table-names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) to the same value on all group members. A setting of 1 is correct for the use of the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) storage engine, which is required for Group Replication. Note that this setting is not the default on all platforms.

**Multithreaded Appliers.** Group Replication members can be configured as multithreaded replicas, enabling transactions to be applied in parallel. A nonzero value for [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) enables the multithreaded applier on the member, and up to 1024 parallel applier threads can be specified. If you do this, the following settings are also required:

[**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order)

This setting is required to ensure that the final commit of parallel transactions is in the same order as the original transactions. Group Replication relies on consistency mechanisms built around the guarantee that all participating members receive and apply committed transactions in the same order.

[**slave\_parallel\_type=LOGICAL\_CLOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type)

This setting is required with [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order). It specifies the policy used to decide which transactions are allowed to execute in parallel on the replica.

Setting [**slave\_parallel\_workers=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) disables parallel execution and gives the replica a single applier thread and no coordinator thread. With that setting, the [**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type) and [**slave\_preserve\_commit\_order**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) options have no effect and are ignored.

### 18.3.2 Group Replication Limitations

The following known limitations exist for Group Replication. Note that the limitations and issues described for multi-primary mode groups can also apply in single-primary mode clusters during a failover event, while the newly elected primary flushes out its applier queue from the old primary.

Tip

Group Replication is built on GTID based replication, therefore you should also be aware of [Section 17.1.3.7, “Restrictions on Replication with GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-restrictions).

**--upgrade=MINIMAL option.** Group Replication cannot be started following a MySQL Server upgrade that uses the MINIMAL option (**--upgrade=MINIMAL**), which does not upgrade system tables on which the replication internals depend.

**Gap Locks.** Group Replication's certification process for concurrent transactions does not take into account [gap locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_gap_lock), as information about gap locks is not available outside of [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html). See [Gap Locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-gap-locks) for more information.

**Note**

For a group in multi-primary mode, unless you rely on [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read) semantics in your applications, we recommend using the [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed) isolation level with Group Replication. InnoDB does not use gap locks in [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed), which aligns the local conflict detection within InnoDB with the distributed conflict detection performed by Group Replication. For a group in single-primary mode, only the primary accepts writes, so the [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed) isolation level is not important to Group Replication.

**Table Locks and Named Locks.** The certification process does not take into account table locks (see [Section 13.3.6, “LOCK TABLES and UNLOCK TABLES Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables)) or named locks (see [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock)).

**Binary Log Checksums.** Up to and including MySQL 8.0.20, Group Replication cannot make use of checksums and does not support their presence in the binary log, so you must set [**binlog\_checksum=NONE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_checksum) when configuring a server instance to become a group member. From MySQL 8.0.21, Group Replication supports checksums, so group members may use the default setting [**binlog\_checksum=CRC32**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_checksum). The setting for [**binlog\_checksum**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_checksum) does not have to be the same for all members of a group.

When checksums are available, Group Replication does not use them to verify incoming events on the **group\_replication\_applier** channel, because events are written to that relay log from multiple sources and before they are actually written to the originating server's binary log, which is when a checksum is generated. Checksums are used to verify the integrity of events on the **group\_replication\_recovery** channel and on any other replication channels on group members.

**SERIALIZABLE Isolation Level.** [**SERIALIZABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_serializable) isolation level is not supported in multi-primary groups by default. Setting a transaction isolation level to **SERIALIZABLE** configures Group Replication to refuse to commit the transaction.

**Concurrent DDL versus DML Operations.** Concurrent data definition statements and data manipulation statements executing against the same object but on different servers is not supported when using multi-primary mode. During execution of Data Definition Language (DDL) statements on an object, executing concurrent Data Manipulation Language (DML) on the same object but on a different server instance has the risk of conflicting DDL executing on different instances not being detected.

**Foreign Keys with Cascading Constraints.** Multi-primary mode groups (members all configured with [**group\_replication\_single\_primary\_mode=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_single_primary_mode)) do not support tables with multi-level foreign key dependencies, specifically tables that have defined **CASCADING** [foreign key constraints](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_foreign_key_constraint). This is because foreign key constraints that result in cascading operations executed by a multi-primary mode group can result in undetected conflicts and lead to inconsistent data across the members of the group. Therefore we recommend setting [**group\_replication\_enforce\_update\_everywhere\_checks=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_enforce_update_everywhere_checks) on server instances used in multi-primary mode groups to avoid undetected conflicts.

In single-primary mode this is not a problem as it does not allow concurrent writes to multiple members of the group and thus there is no risk of undetected conflicts.

**Multi-primary Mode Deadlock.** When a group is operating in multi-primary mode, **SELECT .. FOR UPDATE** statements can result in a deadlock. This is because the lock is not shared across the members of the group, therefore the expectation for such a statement might not be reached.

**Replication Filters.** Global replication filters cannot be used on a MySQL server instance that is configured for Group Replication, because filtering transactions on some servers would make the group unable to reach agreement on a consistent state. Channel specific replication filters can be used on replication channels that are not directly involved with Group Replication, such as where a group member also acts as a replica to a source that is outside the group. They cannot be used on the **group\_replication\_applier** or **group\_replication\_recovery** channels.

**Encrypted Connections.** Support for the TLSv1.3 protocol is available in MySQL Server as of MySQL 8.0.16, provided that MySQL was compiled using OpenSSL 1.1.1 or higher. In MySQL 8.0.16 and MySQL 8.0.17, if the server supports TLSv1.3, the protocol is not supported in the group communication engine and cannot be used by Group Replication. Group Replication supports TLSv1.3 from MySQL 8.0.18, where it can be used for group communication connections and distributed recovery connections.

In MySQL 8.0.18, TLSv1.3 can be used in Group Replication for the distributed recovery connection, but the [**group\_replication\_recovery\_tls\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_tls_version) and [**group\_replication\_recovery\_tls\_ciphersuites**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_tls_ciphersuites) system variables are not available. The donor servers must therefore permit the use of at least one TLSv1.3 ciphersuite that is enabled by default, as listed in [Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#encrypted-connection-protocols-ciphers). From MySQL 8.0.19, you can use the options to configure client support for any selection of ciphersuites, including only non-default ciphersuites if you want.

**Cloning Operations.** Group Replication initiates and manages cloning operations for distributed recovery, but group members that have been set up to support cloning may also participate in cloning operations that a user initiates manually. In releases before MySQL 8.0.20, you cannot initiate a cloning operation manually if the operation involves a group member on which Group Replication is running. From MySQL 8.0.20, you can do this, provided that the cloning operation does not remove and replace the data on the recipient. The statement to initiate the cloning operation must therefore include the **DATA DIRECTORY** clause if Group Replication is running. See [Section 18.5.3.2.4, “Cloning for Other Purposes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-cloning-manual).

#### Limit on Group Size

The maximum number of MySQL servers that can be members of a single replication group is 9. If further members attempt to join the group, their request is refused. This limit has been identified from testing and benchmarking as a safe boundary where the group performs reliably on a stable local area network.

#### Limits on Transaction Size

If an individual transaction results in message contents which are large enough that the message cannot be copied between group members over the network within a 5-second window, members can be suspected of having failed, and then expelled, just because they are busy processing the transaction. Large transactions can also cause the system to slow due to problems with memory allocation. To avoid these issues use the following mitigations:

If unnecessary expulsions occur due to large messages, use the system variable [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) to allow additional time before a member under suspicion of having failed is expelled. You can allow up to an hour after the initial 5-second detection period before a suspect member is expelled from the group. From MySQL 8.0.21, an additional 5 seconds is allowed by default.

Where possible, try and limit the size of your transactions before they are handled by Group Replication. For example, split up files used with **LOAD DATA** into smaller chunks.

Use the system variable [**group\_replication\_transaction\_size\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_transaction_size_limit) to specify a maximum transaction size that the group accepts. In MySQL 8.0, this system variable defaults to a maximum transaction size of 150000000 bytes (approximately 143 MB). Transactions above this size are rolled back and are not sent to Group Replication's Group Communication System (GCS) for distribution to the group. Adjust the value of this variable depending on the maximum message size that you need the group to tolerate, bearing in mind that the time taken to process a transaction is proportional to its size.

Use the system variable [**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold) to specify a message size above which compression is applied. This system variable defaults to 1000000 bytes (1 MB), so large messages are automatically compressed. Compression is carried out by Group Replication's Group Communication System (GCS) when it receives a message that was permitted by the [**group\_replication\_transaction\_size\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_transaction_size_limit) setting but exceeds the [**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold) setting. For more information, see [Section 18.7.3, “Message Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-message-compression).

Use the system variable [**group\_replication\_communication\_max\_message\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_communication_max_message_size) to specify a message size above which messages are fragmented. This system variable defaults to 10485760 bytes (10 MiB), so large messages are automatically fragmented. GCS carries out fragmentation after compression if the compressed message still exceeds the [**group\_replication\_communication\_max\_message\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_communication_max_message_size) limit. In order for a replication group to use fragmentation, all group members must be at MySQL 8.0.16 or above, and the Group Replication communication protocol version in use by the group must allow fragmentation. For more information, see [Section 18.7.4, “Message Fragmentation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-performance-message-fragmentation).

The maximum transaction size, message compression, and message fragmentation can all be deactivated by specifying a zero value for the relevant system variable. If you have deactivated all these safeguards, the upper size limit for a message that can be handled by the applier thread on a member of a replication group is the value of the member's [**slave\_max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_max_allowed_packet) system variable, which has a default and maximum value of 1073741824 bytes (1 GB). A message that exceeds this limit fails when the receiving member attempts to handle it. The upper size limit for a message that a group member can originate and attempt to transmit to the group is 4294967295 bytes (approximately 4 GB). This is a hard limit on the packet size that is accepted by the group communication engine for Group Replication (XCom, a Paxos variant), which receives messages after GCS has handled them. A message that exceeds this limit fails when the originating member attempts to broadcast it.

## 18.4 Monitoring Group Replication

[18.4.1 Group Replication Server States](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-server-states)

[18.4.2 The replication\_group\_members Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-replication-group-members)

[18.4.3 The replication\_group\_member\_stats Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-replication-group-member-stats)

Use the Perfomance Schema tables to monitor Group Replication, assuming that the [Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema-quick-start) is enabled. Group Replication adds the following tables:

[**performance\_schema.replication\_group\_member\_stats**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-member-stats-table)

[**performance\_schema.replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table)

These Perfomance Schema replication tables also show information about Group Replication:

[**performance\_schema.replication\_connection\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-status-table) shows information regarding Group Replication, for example the transactions that have been received from the group and queued in the applier queue (the relay log).

[**performance\_schema.replication\_applier\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-table) shows the state of the Group Replication related channels and threads If there are many different worker threads applying transactions, then the worker tables can also be used to monitor what each worker thread is doing.

The replication channels created by the Group Replication plugin are named:

**group\_replication\_recovery** - This channel is used for the replication changes that are related to the distributed recovery phase.

**group\_replication\_applier** - This channel is used for the incoming changes from the group. This is the channel used to apply transactions coming directly from the group.

From MySQL 8.0.21, Group Replication lifecycle events that are non-error situations are classified as system messages, and are always logged to the server's error log on a replication group member. You can use this information to review the history of the server's membership in a replication group. In previous releases, Group Replication lifecycle events that are non-error situations are classified as information messages, which can be added to the error log by specifying a [**log\_error\_verbosity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_log_error_verbosity) level of 3 for the server.

Some lifecycle events that affect the whole group are logged on every group member, such as a new member entering **ONLINE** status in the group or a primary election. Other events are logged only on the member where they take place, such as super read only mode being enabled or disabled on the member, or the member leaving the group. A number of lifecycle events that can indicate an issue if they occur frequently are logged as warning messages, including the member becoming unreachable and reachable again, and the member starting distributed recovery by state transfer from the binary log or by a remote cloning operation.

The following sections describe how to interpret the monitoring information available for Group Replication.

### 18.4.1 Group Replication Server States

There are various states that a server instance can be in. If servers are communicating properly, all report the same states for all servers. However, if there is a network partition, or a server leaves the group, then different information could be reported, depending on which server is queried. If the server has left the group then it cannot report updated information about the other servers' states. If there is a partition, such that quorum is lost, servers are not able to coordinate between themselves. As a consequence, they cannot guess what the status of different servers is. Therefore, instead of guessing their state they report that some servers are unreachable.

**Table 18.1 Server State**

| **Field** | **Description** | **Group Synchronized** |
| --- | --- | --- |
| **ONLINE** | The member is ready to serve as a fully functional group member, meaning that the client can connect and start executing transactions. | Yes |
| **RECOVERING** | The member is in the process of becoming an active member of the group and is currently going through the recovery process, receiving state transfer from a donor. | No |
| **OFFLINE** | The Group Replication plugin is loaded but the member does not belong to any group. | No |
| **ERROR** | The member is in an error state and is not functioning correctly as a group member. Depending on the exit action set by [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action), the member is in read-only mode ([**super\_read\_only=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only)) and could also be in offline mode ([**offline\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_offline_mode)). Note that a server in offline mode following the **OFFLINE\_MODE** exit action is displayed with **ERROR** status, not **OFFLINE**. A server with the exit action **ABORT\_SERVER** shuts down and is removed from the view of the group. | No |
| **UNREACHABLE** | Whenever the local failure detector suspects that a given server is not reachable, because for example it was disconnected involuntarily, it shows that server's state as **UNREACHABLE**. | No |

### 18.4.2 The replication\_group\_members Table

The [**performance\_schema.replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table) table is used for monitoring the status of the different server instances that are members of the group. The information in the table is updated whenever there is a view change, for example when the configuration of the group is dynamically changed when a new member joins. At that point, servers exchange some of their metadata to synchronize themselves and continue to cooperate together. The information is shared between all the server instances that are members of the replication group, so information on all the group members can be queried from any member. This table can be used to get a high level view of the state of a replication group, for example by issuing:

**SELECT \* FROM performance\_schema.replication\_group\_members;**

+---------------------------+--------------------------------------+--------------+-------------+--------------+-------------+----------------+

| CHANNEL\_NAME | MEMBER\_ID | MEMBER\_HOST | MEMBER\_PORT | MEMBER\_STATE | MEMBER\_ROLE | MEMBER\_VERSION |

+---------------------------+--------------------------------------+--------------+-------------+--------------+-------------+----------------+

| group\_replication\_applier | 041f26d8-f3f3-11e8-adff-080027337932 | example1 | 3306 | ONLINE | SECONDARY | 8.0.13 |

| group\_replication\_applier | f60a3e10-f3f2-11e8-8258-080027337932 | example2 | 3306 | ONLINE | PRIMARY | 8.0.13 |

| group\_replication\_applier | fc890014-f3f2-11e8-a9fd-080027337932 | example3 | 3306 | ONLINE | SECONDARY | 8.0.13 |

+---------------------------+--------------------------------------+--------------+-------------+--------------+-------------+----------------+

Based on this result we can see that the group consists of three members, each member's host and port number which clients use to connect to the member, and the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) of the member. The **MEMBER\_STATE** column shows one of the [Section 18.4.1, “Group Replication Server States”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-server-states), in this case it shows that all three members in this group are **ONLINE**, and the **MEMBER\_ROLE** column shows that there are two secondaries, and a single primary. Therefore this group must be running in single-primary mode. The **MEMBER\_VERSION** column can be useful when you are upgrading a group and are combining members running different MySQL versions. See [Section 18.4.1, “Group Replication Server States”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-server-states) for more information.

For more information about the **Member\_host** value and its impact on the distributed recovery process, see [Section 18.2.1.3, “User Credentials For Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-user-credentials).

### 18.4.3 The replication\_group\_member\_stats Table

Each member in a replication group certifies and applies transactions received by the group. Statistics regarding the certifier and applier procedures are useful to understand how the applier queue is growing, how many conflicts have been found, how many transactions were checked, which transactions are committed everywhere, and so on.

The [**performance\_schema.replication\_group\_member\_stats**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-member-stats-table) table provides group-level information related to the certification process, and also statistics for the transactions received and originated by each individual member of the replication group. The information is shared between all the server instances that are members of the replication group, so information on all the group members can be queried from any member. Note that refreshing of statistics for remote members is controlled by the message period specified in the [**group\_replication\_flow\_control\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_period) option, so these can differ slightly from the locally collected statistics for the member where the query is made. To use this table to monitor a Group Replication member, issue:

mysql> **SELECT \* FROM performance\_schema.replication\_group\_member\_stats\G**

These fields are important for monitoring the performance of the members connected in the group. For example, suppose that one of the group’s members always reports a large number of transactions in its queue compared to other members. This means that the member is delayed and is not able to keep up to date with the other members of the group. Based on this information, you could decide to either remove the member from the group, or delay the processing of transactions on the other members of the group in order to reduce the number of queued transactions. This information can also help you to decide how to adjust the flow control of the Group Replication plugin, see [Section 18.7.2, “Flow Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-flow-control).

## 18.5 Group Replication Operations

[18.5.1 Configuring an Online Group](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-configuring-online-group)

[18.5.2 Transaction Consistency Guarantees](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-consistency-guarantees)

[18.5.3 Distributed Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery)

[18.5.4 Network Partitioning](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-network-partitioning)

[18.5.5 Support For IPv6 And For Mixed IPv6 And IPv4 Groups](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-ipv6)

[18.5.6 Using MySQL Enterprise Backup with Group Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-enterprise-backup)

This section explains common operations for managing groups.

### 18.5.1 Configuring an Online Group

[18.5.1.1 Changing a Group's Primary Member](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-changing-primary-member)

[18.5.1.2 Changing a Group's Mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-changing-group-mode)

[18.5.1.3 Using Group Replication Group Write Consensus](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-group-write-consensus)

[18.5.1.4 Setting a Group's Communication Protocol Version](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-communication-protocol)

You can configure an online group while Group Replication is running by using a set of UDFs, which rely on a group action coordinator. These UDFs are installed by the Group Replication plugin in version 8.0.13 and higher. This section describes how changes are made to a running group, and the available UDFs.

**Important**

For the coordinator to be able to configure group wide actions on a running group, all members must be running MySQL 8.0.13 or higher and have the UDFs installed.

To use the UDFs, connect to a member of the running group and issue the UDF with the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement. The Group Replication plugin processes the action and its parameters and the coordinator sends it to all members which are visible to the member where you issued the UDF. If the action is accepted, all members execute the action and send a termination message when completed. Once all members declare the action as finished, the invoking member returns the result to the client.

When configuring a whole group, the distributed nature of the operations means that they interact with many processes of the Group Replication plugin, and therefore you should observe the following:

**You can issue configuration operations everywhere.** If you want to make member A the new primary you do not need to invoke the operation on member A. All operations are sent and executed in a coordinated way on all group members. Also, this distributed execution of an operation has a different ramification: if the invoking member dies, any already running configuration process continues to run on other members. In the unlikely event that the invoking member dies, you can still use the monitoring features to ensure other members complete the operation successfully.

**All members must be online.** To simplify the migration or election processes and guarantee they are as fast as possible, the group must not contain any member currently in the distributed recovery process, otherwise the configuration action is rejected by the member where you issue the statement.

**No members can join a group during a configuration change.** Any member that attempts to join the group during a coordinated configuration change leaves the group and cancels its join process.

**Only one configuration at once.** A group which is executing a configuration change cannot accept any other group configuration change, because concurrent configuration operations could lead to member divergence.

**All members must be running MySQL 8.0.13 or higher.** Due to the distributed nature of the configuration actions, all members must recognize them in order to execute them. The operation is therefore rejected if any server running MySQL Server version 8.0.12 or lower is present in the group.

#### 18.5.1.1 Changing a Group's Primary Member

This section explains how to change which member of a single-primary group is the primary. The function used to change a group's mode can be run on any member.

##### Changing which Member is Primary

Use the [**group\_replication\_set\_as\_primary()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-as-primary) UDF to change which member is the primary in a single-primary group. This function has no effect if issued on a member of a multi-primary group.

**Note**

Only a primary member can receive data from an external source and write it to the group. If a standard source to replica replication channel is running on the existing primary member in addition to the Group Replication channels, you can only change the primary member after that replication channel is stopped.

If the group is waiting on any uncommitted transactions, the operation cannot complete until they are committed, rolled back, or terminated.

If you issue the UDF on a member running a MySQL Server version from 8.0.17, and all members are running MySQL Server version 8.0.17 or higher, you can only specify a new primary member that is running the lowest MySQL Server version in the group, based on the patch version. This safeguard is applied to ensure the group maintains compatibility with new functions. If any member is running a MySQL Server version between MySQL 8.0.13 and MySQL 8.0.16, this safeguard is not enforced for the group and you can specify any new primary member, but it is recommended to select a primary that is running the lowest MySQL Server version in the group.

Pass in the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) of the member which you want to become the new primary of the group by issuing:

**SELECT group\_replication\_set\_as\_primary(member\_uuid);**

While the action runs, you can check its progress by issuing:

**SELECT event\_name, work\_completed, work\_estimated FROM performance\_schema.events\_stages\_current WHERE event\_name LIKE "%stage/group\_rpl%";**

+----------------------------------------------------------------------------------+----------------+----------------+

| event\_name | work\_completed | work\_estimated |

+----------------------------------------------------------------------------------+----------------+----------------+

| stage/group\_rpl/Primary Election: Waiting for members to turn on super\_read\_only | 3 | 5 |

+----------------------------------------------------------------------------------+----------------+----------------+

#### 18.5.1.2 Changing a Group's Mode

This section explains how to change the mode which a group is running in, either single or multi-primary. The functions used to change a group's mode can be run on any member.

##### Changing to Single-Primary Mode

Use the [**group\_replication\_switch\_to\_single\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-single-primary-mode) UDF to change a group running in multi-primary mode to single-primary mode by issuing:

**SELECT group\_replication\_switch\_to\_single\_primary\_mode()**

When you change to single-primary mode, strict consistency checks are also disabled on all group members, as required in single-primary mode ([**group\_replication\_enforce\_update\_everywhere\_checks=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_enforce_update_everywhere_checks)).

If no string is passed in, the election of the new primary in the resulting single-primary group follows the election policies described in [Section 18.1.3.1, “Single-Primary Mode”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-single-primary-mode). To override the election process and configure a specific member of the multi-primary group as the new primary in the process, get the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) of the member and pass it to [**group\_replication\_switch\_to\_single\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-single-primary-mode). For example issue:

**SELECT group\_replication\_switch\_to\_single\_primary\_mode(*member\_uuid*);**

If you issue the UDF on a member running a MySQL Server version from 8.0.17, and all members are running MySQL Server version 8.0.17 or higher, you can only specify a new primary member that is running the lowest MySQL Server version in the group, based on the patch version. This safeguard is applied to ensure the group maintains compatibility with new functions. If you do not specify a new primary member, the election process considers the patch version of the group members.

If any member is running a MySQL Server version between MySQL 8.0.13 and MySQL 8.0.16, this safeguard is not enforced for the group and you can specify any new primary member, but it is recommended to select a primary that is running the lowest MySQL Server version in the group. If you do not specify a new primary member, the election process considers only the major version of the group members.

While the action runs, you can check its progress by issuing:

**SELECT event\_name, work\_completed, work\_estimated FROM performance\_schema.events\_stages\_current WHERE event\_name LIKE "%stage/group\_rpl%";**

+----------------------------------------------------------------------------+----------------+----------------+

| event\_name | work\_completed | work\_estimated |

+----------------------------------------------------------------------------+----------------+----------------+

| stage/group\_rpl/Primary Switch: waiting for pending transactions to finish | 4 | 20 |

+----------------------------------------------------------------------------+----------------+----------------+

##### Changing to Multi-Primary Mode

Use the [**group\_replication\_switch\_to\_multi\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-multi-primary-mode) UDF to change a group running in single-primary mode to multi-primary mode by issuing:

**SELECT group\_replication\_switch\_to\_multi\_primary\_mode()**

After some coordinated group operations to ensure the safety and consistency of your data, all members which belong to the group become primaries.

When you change a group that was running in single-primary mode to run in multi-primary mode, members running MySQL 8.0.17 or higher are automatically placed in read-only mode if they are running a higher MySQL server version than the lowest version present in the group. Members running MySQL 8.0.16 or lower do not carry out this check, and are always placed in read-write mode.

While the action runs, you can check its progress by issuing:

SELECT event\_name, work\_completed, work\_estimated FROM performance\_schema.events\_stages\_current WHERE event\_name LIKE "%stage/group\_rpl%";

+----------------------------------------------------------------------+----------------+----------------+

| event\_name | work\_completed | work\_estimated |

+----------------------------------------------------------------------+----------------+----------------+

| stage/group\_rpl/Multi-primary Switch: applying buffered transactions | 0 | 1 |

+----------------------------------------------------------------------+----------------+----------------+

#### 18.5.1.3 Using Group Replication Group Write Consensus

This section explains how to inspect and configure the maximum number of consensus instances at any time for a group. This maximum is referred to as the event horizon for a group, and is the maximum number of consensus instances that the group can execute in parallel. This enables you to fine tune the performance of your Group Replication deployment. For example, the default value of 10 is suitable for a group running on a LAN, but for groups operating over a slower network such as a WAN, increase this number to improve performance.

##### Inspecting a Group's Write Concurrency

Use the [**group\_replication\_get\_write\_concurrency()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-get-write-concurrency) UDF to inspect a group's event horizon value at runtime by issuing:

**SELECT group\_replication\_get\_write\_concurrency();**

##### Configuring a Group's Write Concurrency

Use the [**group\_replication\_set\_write\_concurrency()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-write-concurrency) UDF to set the maximum number of consensus instances that the system can execute in parallel by issuing:

**SELECT group\_replication\_set\_write\_concurrency(*instances*);**

where ***instances*** is the new maximum number of consensus instances. The [**GROUP\_REPLICATION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_group-replication-admin) privilege is required to use this UDF.

#### 18.5.1.4 Setting a Group's Communication Protocol Version

From MySQL 8.0.16, Group Replication has the concept of a communication protocol for the group. The Group Replication communication protocol version can be managed explicitly, and set to accommodate the oldest MySQL Server version that you want the group to support. This enables groups to be formed from members at different MySQL Server versions while ensuring backward compatibility. Versions from MySQL 5.7.14 allow compression of messages, and versions from MySQL 8.0.16 also allow fragmentation of messages. All members of the group must use the same communication protocol version, so that group members can be at different MySQL Server releases but only send messages that can be understood by all group members.

A MySQL server at version X can only join and reach **ONLINE** status in a replication group if the group's communication protocol version is less than or equal to X. When a new member joins a replication group, it checks the communication protocol version that is announced by the existing members of the group. If the joining member supports that version, it joins the group and uses the communication protocol that the group has announced, even if the member supports additional communication capabilities. If the joining member does not support the communication protocol version, it is expelled from the group.

If two members attempt to join in the same membership change event, they can only join if the communication protocol version for both members is already compatible with the group's communication protocol version. Members with different communication protocol versions from the group must join in isolation. For example:

One MySQL Server 8.0.16 instance can successfully join a group that uses the communication protocol version 5.7.24.

One MySQL Server 5.7.24 instance cannot successfully join a group that uses the communication protocol version 8.0.16.

Two MySQL Server 8.0.16 instances cannot simultaneously join a group that uses the communication protocol version 5.7.24.

Two MySQL Server 8.0.16 instances can simultaneously join a group that uses the communication protocol version 8.0.16.

You can inspect the communication protocol in use by a group by using the [**group\_replication\_get\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-get-communication-protocol) UDF, which returns the oldest MySQL Server version that the group supports. All existing members of the group return the same communication protocol version. For example:

SELECT group\_replication\_get\_communication\_protocol();

+------------------------------------------------+

| group\_replication\_get\_communication\_protocol() |

+------------------------------------------------+

| 8.0.16 |

+------------------------------------------------+

Note that the [**group\_replication\_get\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-get-communication-protocol) UDF returns the minimum MySQL version that the group supports, which might differ from the version number that was passed to the [**group\_replication\_set\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-communication-protocol) UDF, and from the MySQL Server version that is installed on the member where you use the UDF.

If you need to change the communication protocol version of a group so that members at earlier releases can join, use the [**group\_replication\_set\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-communication-protocol) UDF to specify the MySQL Server version of the oldest member that you want to allow. This makes the group fall back to a compatible communication protocol version if possible. The [**GROUP\_REPLICATION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_group-replication-admin) privilege is required to use this UDF, and all existing group members must be online when you issue the statement, with no loss of majority. For example:

**SELECT group\_replication\_set\_communication\_protocol("5.7.25");**

If you upgrade all the members of a replication group to a new MySQL Server release, the group's communication protocol version is not automatically upgraded to match. If you no longer need to support members at earlier releases, you can use the [**group\_replication\_set\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-communication-protocol) UDF to set the communication protocol version to the new MySQL Server version to which you have upgraded the members. For example:

**SELECT group\_replication\_set\_communication\_protocol("8.0.16");**

The [**group\_replication\_set\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-communication-protocol) UDF is implemented as a group action, so it is executed at the same time on all members of the group. The group action starts buffering messages and waits for delivery of any outgoing messages that were already in progress to complete, then changes the communication protocol version and sends the buffered messages. If a member attempts to join the group at any time after you change the communication protocol version, the group members announce the new protocol version.

MySQL InnoDB cluster automatically and transparently manages the communication protocol versions of its members, whenever the cluster topology is changed using AdminAPI operations. An InnoDB cluster always uses the most recent communication protocol version that is supported by all the instances that are currently part of the cluster or joining it. For details, see [InnoDB Cluster and Group Replication Protocol](https://dev.mysql.com/doc/mysql-shell/8.0/en/monitoring-innodb-cluster.html#innodb-cluster-group-replication-protocol).

### 18.5.2 Transaction Consistency Guarantees

[18.5.2.1 Understanding Transaction Consistency Guarantees](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-understanding-consistency-guarantees)

[18.5.2.2 Configuring Transaction Consistency Guarantees](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-configuring-consistency-guarantees)

One of the major implications of a distributed system such as Group Replication is the consistency guarantees that it provides as a group. In other words, the consistency of the global synchronization of transactions distributed across the members of the group. This section describes how Group Replication handles consistency guarantees depending on the events that occur in a group, and how to best configure your group's consistency guarantees.

#### 18.5.2.1 Understanding Transaction Consistency Guarantees

In terms of distributed consistency guarantees, either in normal or failure repair operations, Group Replication has always been an eventual consistency system. This means that as soon as the incoming traffic slows down or stops, all group members have the same data content. The events that relate to the consistency of a system can be split into control operations, either manual or automatically triggered by failures; and data flow operations.

For Group Replication, the control operations that can be evaluated in terms of consistency are:

a member joining or leaving, which is covered by Group Replication's [Section 18.5.3, “Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery) and write protection.

network failures, which are covered by the fencing modes.

in single-primary groups, primary failover, which can also be an operation triggered by [**group\_replication\_set\_as\_primary()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-as-primary).

##### Consistency Guarantees and Primary Failover

In a single-primary group, in the event of a primary failover when a secondary is promoted to primary, the new primary can either be made available to application traffic immediately, regardless of how large the replication backlog is, or alternatively access to it can be restricted until the backlog has been applied.

With the first approach, the group takes the minimum time possible to secure a stable group membership after a primary failure by electing a new primary and then allowing data access immediately while it is still applying any possible backlog from the old primary. Write consistency is ensured, but reads can temporarily retrieve stale data while the new primary applies the backlog. For example, if client C1 wrote **A=2 WHERE A=1** on the old primary just before its failure, when client C1 is reconnected to the new primary it could potentially read **A=1** until the new primary applies its backlog and catches up with the state of the old primary before it left the group.

With the second alternative, the system secures a stable group membership after the primary failure and elects a new primary in the same way as the first alternative, but in this case the group then waits until the new primary applies all backlog and only then does it permit data access. This ensures that in a situation as described previously, when client C1 is reconnected to the new primary it reads **A=2**. However, the trade-off is that the time required to failover is then proportional to the size of the backlog, which on a correctly configured group should be small .

Prior to MySQL 8.0.14 there was no way to configure the failover policy, by default availability was maximized as described in the first approach. In a group with members running MySQL 8.0.14 and higher, you can configure the level of transaction consistency guarantees provided by members during primary failover using the [**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) variable. See [Impact of Consistency on Primary Election](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-consistency-level-impact-election).

##### Data Flow Operations

Data flow is relevant to group consistency guarantees due to the reads and writes executed against a group, especially when these operations are distributed across all members. Data flow operations apply to both modes of Group Replication: single-primary and multi-primary, however to make this explanation clearer it is restricted to single-primary mode. The usual way to split incoming read or write transactions across a single-primary group's members is to route writes to the primary and evenly distribute reads to the secondaries. Since the group should behave as a single entity, it is reasonable to expect that writes on the primary are instantaneously available on the secondaries. Although Group Replication is written using Group Communication System (GCS) protocols that implement the Paxos algorithm, some parts of Group Replication are asynchronous, which implies that data is asynchronously applied to secondaries. This means that a client C2 can write **B=2 WHERE B=1** on the primary, immediately connect to a secondary and read **B=1**. This is because the secondary is still applying backlog, and has not applied the transaction which was applied by the primary.

##### Transaction Synchronization Points

You configure a group's consistency guarantee based on the point at which you want to synchronize transactions across the group. To help you understand the concept, this section simplifies the points of synchronizing transactions across a group to be at the time of a read operation or at the time of a write operation. If data is synchronized at the time of a read, the current client session waits until a given point, which is the point in time that all preceding update transactions have been applied, before it can start executing. With this approach, only this session is affected, all other concurrent data operations are not affected.

If data is synchronized at the time of write, the writing session waits until all secondaries have written their data. Group Replication uses a total order on writes, and therefore this implies waiting for this and all preceding writes that are in secondaries’ queues to be applied. Therefore when using this synchronization point, the writing session waits for all secondaries queues to be applied.

Any alternative ensures that in the situation described for client C2 would always read **B=2** even if immediately connected to a secondary. Each alternative has its advantages and disadvantages, which are directly related to your system workload. The following examples describe different types of workloads and advise which point of synchronization is appropriate.

Imagine the following situations:

you want to load balance your reads without deploying additional restrictions on which server you read from to avoid reading stale data, group writes are much less common than group reads.

you have a group that has a predominantly read-only data, you want read-write transactions to be applied everywhere once they commit, so that subsequent reads are done on up-to-date data that includes the latest write. This ensures that you do not pay the synchronization cost for every RO transaction, but only on RW ones.

In these cases, you should choose to synchronize on writes.

Imagine the following situations:

you want to load balance your reads without deploying additional restrictions on which server you read from to avoid reading stale data, group writes are much more common than group reads.

you want specific transactions in your workload to always read up-to-date data from the group, for example whenever sensitive data is updated (such as credentials for a file or similar data) and you want to enforce that reads retrieve the most up to date value.

In these cases, you should choose to synchronize on reads.

#### 18.5.2.2 Configuring Transaction Consistency Guarantees

Although the [Transaction Synchronization Points](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-synchronization-points) section explains that conceptually there are two synchronization points from which you can choose: on read or on write, these terms were a simplification and the terms used in Group Replication are: before and after transaction execution. The consistency level can have a different impact on read-only (RO) and read-write (RW) transactions processed by the group as demonstrated in this section.

[How to Choose a Consistency Level](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-choose-consistency-level)

[Impacts of Consistency Levels](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-consistency-level-impacts)

[Impact of Consistency on Primary Election](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-consistency-level-impact-election)

The following list shows the possible consistency levels that you can configure in Group Replication using the [**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) variable, in order of increasing transaction consistency guarantee:

**EVENTUAL**

Both RO and RW transactions do not wait for preceding transactions to be applied before executing. This was the behavior of Group Replication before the [**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) variable was added. A RW transaction does not wait for other members to apply a transaction. This means that a transaction could be externalized on one member before the others. This also means that in the event of a primary failover, the new primary can accept new RO and RW transactions before the previous primary transactions are all applied. RO transactions could result in outdated values, RW transactions could result in a rollback due to conflicts.

**BEFORE\_ON\_PRIMARY\_FAILOVER**

New RO or RW transactions with a newly elected primary that is applying backlog from the old primary are held (not applied) until any backlog has been applied. This ensures that when a primary failover happens, intentionally or not, clients always see the latest value on the primary. This guarantees consistency, but means that clients must be able to handle the delay in the event that a backlog is being applied. Usually this delay should be minimal, but it does depend on the size of the backlog.

**BEFORE**

A RW transaction waits for all preceding transactions to complete before being applied. A RO transaction waits for all preceding transactions to complete before being executed. This ensures that this transaction reads the latest value by only affecting the latency of the transaction. This reduces the overhead of synchronization on every RW transaction, by ensuring synchronization is used only on RO transactions. This consistency level also includes the consistency guarantees provided by **BEFORE\_ON\_PRIMARY\_FAILOVER**.

**AFTER**

A RW transaction waits until its changes have been applied to all of the other members. This value has no effect on RO transactions. This mode ensures that when a transaction is committed on the local member, any subsequent transaction reads the written value or a more recent value on any group member. Use this mode with a group that is used for predominantly RO operations to ensure that applied RW transactions are applied everywhere once they commit. This could be used by your application to ensure that subsequent reads fetch the latest data which includes the latest writes. This reduces the overhead of synchronization on every RO transaction, by ensuring synchronization is used only on RW transactions. This consistency level also includes the consistency guarantees provided by **BEFORE\_ON\_PRIMARY\_FAILOVER**.

**BEFORE\_AND\_AFTER**

A RW transaction waits for 1) all preceding transactions to complete before being applied and 2) until its changes have been applied on other members. A RO transaction waits for all preceding transactions to complete before execution takes place. This consistency level also includes the consistency guarantees provided by **BEFORE\_ON\_PRIMARY\_FAILOVER**.

The **BEFORE** and **BEFORE\_AND\_AFTER** consistency levels can be both used on RO and RW transactions. The **AFTER** consistency level has no impact on RO transactions, because they do not generate changes.

##### How to Choose a Consistency Level

The different consistency levels provide flexibility to both DBAs, who can use them to set up their infrastructure; and to developers who can use the consistency level that best suits their application's requirements. The following scenarios show how to choose a consistency guarantee level based on how you use your group:

Scenario 1 you want to load balance your reads without worrying about stale reads, your group write operations are considerably fewer than your group read operations. In this case, you should choose **AFTER**.

Scenario 2 you have a data set that applies a lot of writes and you want to do occasional reads without having to worry about reading stale data. In this case, you should choose **BEFORE**.

Scenario 3 you want specific transactions in your workload to always read up-to-date data from the group, so that whenever that sensitive data is updated (such as credentials for a file or similar data) you want to enforce that reads always read the most up to date value. In this case, you should choose **BEFORE**.

Scenario 4 you have a group that has predominantly read-only (RO) data, you want your read-write (RW) transactions to be applied everywhere once they commit, so that subsequent reads are done on up-to-date data that includes your latest writes and you do not pay the synchronization on every RO transaction, but only on RW ones. In this case, you should choose **AFTER**.

Scenario 5 you have a group that has predominantly read-only data, you want your read-write (RW) transactions to always read up-to-date data from the group and to be applied everywhere once they commit, so that subsequent reads are done on up-to-date data that includes your latest write and you do not pay the synchronization on every read-only (RO) transaction, but only on RW ones. In this case, you should choose **BEFORE\_AND\_AFTER**.

You have the freedom to choose the scope at which the consistency level is enforced. This is important because consistency levels could have a negative impact on group performance if you set them at a global scope. Therefore you can configure the consistency level of a group by using the [**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) system variable at different scopes.

To enforce the consistency level on the current session, use the session scope:

> SET @@SESSION.group\_replication\_consistency= 'BEFORE';

To enforce the consistency level on all sessions, use the global scope:

> SET @@GLOBAL.group\_replication\_consistency= 'BEFORE';

The possibility of setting the consistency level on specific sessions enables you to take advantage of scenarios such as:

Scenario 6 A given system handles several instructions that do not require a strong consistency level, but one kind of instruction does require strong consistency: managing access permissions to documents;. In this scenario, the system changes access permissions and it wants to be sure that all clients see the correct permission. You only need to **SET @@SESSION.group\_replication\_consistency= ‘AFTER’**, on those instructions and leave the other instructions to run with **EVENTUAL** set at the global scope.

Scenario 7 On the same system as described in Scenario 6, every day an instruction needs to do some analytical processing, and as such it requires to always read the most up-to-date data. To achieve this, you only need to **SET @@SESSION.group\_replication\_consistency= ‘BEFORE’** on that specific instruction.

To summarize, you do not need to run all transactions with a specific consistency level, especially if only some transactions actually require it.

Note that all read-write transactions are totally ordered in Group Replication, so even when you set the consistency level to **AFTER** for the current session this transaction waits until its changes are applied on all members, which means waiting for this and all preceding transactions that could be in the secondaries' queues. In practice, the consistency level **AFTER** waits for everything until and including this transaction.

##### Impacts of Consistency Levels

Another way to classify the consistency levels is in terms of impact on the group, that is, the repercussions that the consistency levels have on the other members.

The **BEFORE** consistency level, apart from being ordered on the transaction stream, only impacts on the local member. That is, it does not require coordination with the other members and does not have repercussions on their transactions. In other words, **BEFORE** only impacts the transactions on which it is used.

The **AFTER** and **BEFORE\_AND\_AFTER** consistency levels do have side-effects on concurrent transactions executed on other members. These consistency levels make the other members transactions wait if transactions with the **EVENTUAL** consistency level start while a transaction with **AFTER** or **BEFORE\_AND\_AFTER** is executing. The other members wait until the **AFTER** transaction is committed on that member, even if the other member's transactions have the **EVENTUAL** consistency level. In other words, **AFTER** and **BEFORE\_AND\_AFTER** impact all **ONLINE** group members.

To illustrate this further, imagine a group with 3 members, M1, M2 and M3. On member M1 a client issues:

> SET @@SESSION.group\_replication\_consistency= AFTER;

> BEGIN;

> INSERT INTO t1 VALUES (1);

> COMMIT;

Then, while the above transaction is being applied, on member M2 a client issues:

> SET SESSION group\_replication\_consistency= EVENTUAL;

In this situation, even though the second transaction's consistency level is **EVENTUAL**, because it starts executing while the first transaction is already in the commit phase on M2, the second transaction has to wait for the first transaction to finish the commit and only then can it execute.

You can only use the consistency levels **BEFORE**, **AFTER** and **BEFORE\_AND\_AFTER** on **ONLINE** members, attempting to use them on members in other states causes a session error.

Transactions whose consistency level is not **EVENTUAL** hold execution until a timeout, configured by [**wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_wait_timeout) value is reached, which defaults to 8 hours. If the timeout is reached an **ER\_GR\_HOLD\_WAIT\_TIMEOUT** error is thrown.

##### Impact of Consistency on Primary Election

This section describes how a group's consistency level impacts on a single-primary group that has elected a new primary. Such a group automatically detects failures and adjusts the view of the members that are active, in other words the membership configuration. Furthermore, if a group is deployed in single-primary mode, whenever the group's membership changes there is a check performed to detect if there is still a primary member in the group. If there is none, a new one is selected from the list of secondary members. Typically, this is known as the secondary promotion.

Given the fact that the system detects failures and reconfigures itself automatically, the user may also expect that once the promotion takes place, the new primary is in the exact state, data-wise, as that of the old one. In other words, the user may expect that there is no backlog of replicated transactions to be applied on the new primary once he is able to read from and write to it. In practical terms, the user may expect that once his application fails-over to the new primary, there would be no chance, even if temporarily, to read old data or write into old data records.

When flow control is activated and properly tuned on a group, there is only a small chance of transiently reading stale data from a newly elected primary immediately after the promotion, as there should not be a backlog, or if there is one it should be small. Moreover, you might have a proxy or middleware layers that govern application accesses to the primary after a promotion and enforce the consistency criteria at that level. If your group members are using MySQL 8.0.14 or higher, you can specify the behavior of the new primary once it is promoted using the [**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) variable, which controls whether a newly elected primary blocks both reads and writes until after the backlog is fully applied or if it behaves in the manner of members running MySQL 8.0.13 or earlier. If the [**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) option was set to **BEFORE\_ON\_PRIMARY\_FAILOVER** on a newly elected primary which has backlog to apply, and transactions are issued against the new primary while it is still applying the backlog, incoming transactions are blocked until the backlog is fully applied. Thus, the following anomalies are prevented:

No stale reads for read-only and read-write transactions. This prevents stale reads from being externalized to the application by the new primary.

No spurious roll backs for read-write transactions, due to write-write conflicts with replicated read-write transactions still in the backlog waiting to be applied.

No read skew on read-write transactions, such as:

> BEGIN;

> SELECT x FROM t1; -- x=1 because x=2 is in the backlog;

> INSERT x INTO t2;

> COMMIT;

This query should not cause a conflict but writes outdated values.

To summarize, when [**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) is set to **BEFORE\_ON\_PRIMARY\_FAILOVER** you are choosing to prioritize consistency over availability, because reads and writes are held whenever a new primary is elected. This is the trade-off you have to consider when configuring your group. It should also be remembered that if flow control is working correctly, backlog should be minimal. Note that the higher consistency levels **BEFORE**, **AFTER**, and **BEFORE\_AND\_AFTER** also include the consistency guarantees provided by **BEFORE\_ON\_PRIMARY\_FAILOVER**.

To guarantee that the group provides the same consistency level regardless of which member is promoted to primary, all members of the group should have **BEFORE\_ON\_PRIMARY\_FAILOVER** (or a higher consistency level) persisted to their configuration. For example on each member issue:

> SET PERSIST group\_replication\_consistency='BEFORE\_ON\_PRIMARY\_FAILOVER';

This ensures that the members all behave in the same way, and that the configuration is persisted after a restart of the member.

Although all writes are held when using **BEFORE\_ON\_PRIMARY\_FAILOVER** consistency level, not all reads are blocked to ensure that you can still inspect the server while it is applying backlog after a promotion took place. This is useful for debugging, monitoring, observability and troubleshooting. Some queries that do not modify data are allowed, such as the following:

[**SHOW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show) statements

[**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) statements

[**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) statements

**EMPTY** statements

[**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statements

using [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements against the **performance\_schema** and **sys** databases

using [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements against the **PROCESSLIST** table from the **infoschema** database

[**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements that do not use tables or user defined functions

[**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) statements

[**SHUTDOWN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#shutdown) statements

[**RESET PERSIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist) statements

A transaction cannot be on-hold forever, and if the time held exceeds [**wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_wait_timeout) it returns an ER\_GR\_HOLD\_WAIT\_TIMEOUT error.

### 18.5.3 Distributed Recovery

[18.5.3.1 Connections for Distributed Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-connections)

[18.5.3.2 Cloning for Distributed Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-cloning)

[18.5.3.3 Configuring Distributed Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-tuning-recovery)

[18.5.3.4 Fault Tolerance for Distributed Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-fault)

[18.5.3.5 How Distributed Recovery Works](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-view-changes)

Whenever a member joins or rejoins a replication group, it must catch up with the transactions that were applied by the group members before it joined, or while it was away. This process is called distributed recovery.

The joining member begins by checking the relay log for its **group\_replication\_applier** channel for any transactions that it already received from the group but did not yet apply. If the joining member was in the group previously, it might find unapplied transactions from before it left, in which case it applies these as a first step. A member that is new to the group does not have anything to apply.

After this, the joining member connects to an online existing member to carry out state transfer. The joining member transfers all the transactions that took place in the group before it joined or while it was away, which are provided by the existing member (called the donor). Next, the joining member applies the transactions that took place in the group while this state transfer was in progress. When this process is complete, the joining member has caught up with the remaining servers in the group, and it begins to participate normally in the group.

Group Replication uses a combination of these methods for state transfer during distributed recovery:

A remote cloning operation using the clone plugin's function, which is available from MySQL 8.0.17. To enable this method of state transfer, you must install the clone plugin on the group members and the joining member. Group Replication automatically configures the required clone plugin settings and manages the remote cloning operation.

Replicating from a donor's binary log and applying the transactions on the joining member. This method uses a standard asynchronous replication channel named **group\_replication\_recovery** that is established between the donor and the joining member.

Group Replication automatically selects the best combination of these methods for state transfer after you issue [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) on the joining member. To do this, Group Replication checks which existing members are suitable as donors, how many transactions the joining member needs from a donor, and whether any required transactions are no longer present in the binary log files on any group member. If the transaction gap between the joining member and a suitable donor is large, or if some required transactions are not in any donor's binary log files, Group Replication begins distributed recovery with a remote cloning operation. If there is not a large transaction gap, or if the clone plugin is not installed, Group Replication proceeds directly to state transfer from a donor's binary log.

During a remote cloning operation, the existing data on the joining member is removed, and replaced with a copy of the donor's data. When the remote cloning operation is complete and the joining member has restarted, state transfer from a donor's binary log is carried out to get the transactions that the group applied while the remote cloning operation was in progress.

During state transfer from a donor's binary log, the joining member replicates and applies the required transactions from the donor's binary log, applying the transactions as they are received, up to the point where the binary log records that the joining member joined the group (a view change event). While this is in progress, the joining member buffers the new transactions that the group applies. When state transfer from the binary log is complete, the joining member applies the buffered transactions.

When the joining member is up to date with all the group's transactions, it is declared online and can participate in the group as a normal member, and distributed recovery is complete.

Tip

State transfer from the binary log is Group Replication's base mechanism for distributed recovery, and if the donors and joining members in your replication group are not set up to support cloning, this is the only available option. As state transfer from the binary log is based on classic asynchronous replication, it might take a very long time if the server joining the group does not have the group's data at all, or has data taken from a very old backup image. In this situation, it is therefore recommended that before adding a server to the group, you should set it up with the group's data by transferring a fairly recent snapshot of a server already in the group. This minimizes the time taken for distributed recovery, and reduces the impact on donor servers, since they have to retain and transfer fewer binary log files.

#### 18.5.3.1 Connections for Distributed Recovery

When a joining member connects to an online existing member for state transfer during distributed recovery, the joining member acts as a client on the connection and the existing member acts as a server. When state transfer from the donor's binary log is in progress over this connection (using the asynchronous replication channel **group\_replication\_recovery**), the joining member acts as the replica and the existing member acts as the source. When a remote cloning operation is in progress over this connection, the joining member acts as a recipient and the existing member acts as a donor. Configuration settings that apply to those roles outside the Group Replication context can apply for Group Replication also, unless they are overridden by a Group Replication-specific configuration setting or behavior.

The connection that an existing member offers to a joining member for distributed recovery is not the same connection that is used by Group Replication for communication between online members of the group.

The connection used by the group communication engine for Group Replication (XCom, a Paxos variant) for TCP communication between remote XCom instances is specified by the [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) system variable. This connection is used for TCP/IP messages between online members. Communication with the local instance is over an input channel using shared memory.

For distributed recovery, up to MySQL 8.0.20, group members offer their standard SQL client connection to joining members, as specified by MySQL Server's [**hostname**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_hostname) and [**port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_port) system variables. If an alternative port number is specified by the [**report\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_port) system variable, that one is used instead.

From MySQL 8.0.21, group members may advertise an alternative list of distributed recovery endpoints as dedicated client connections for joining members, allowing you to control distributed recovery traffic separately from connections by regular client users of the member. You specify this list using the [**group\_replication\_advertise\_recovery\_endpoints**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_advertise_recovery_endpoints) system variable, and the member transmits their list of distributed recovery endpoints to the group when they join the group. The default is that the member continues to offer the standard SQL client connection as in earlier releases.

**Important**

Distributed recovery can fail if a joining member cannot correctly identify the other members using the host name as defined by MySQL Server's [**hostname**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_hostname) system variable. It is recommended that operating systems running MySQL have a properly configured unique host name, either using DNS or local settings. The host name that the server is using for SQL client connections can be verified in the **Member\_host** column of the Performance Schema table [**replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table). If multiple group members externalize a default host name set by the operating system, there is a chance of the joining member not resolving it to the correct member address and not being able to connect for distributed recovery. In this situation you can use MySQL Server's [**report\_host**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_host) system variable to configure a unique host name to be externalized by each of the servers.

The steps for a joining member to establish a connection for distributed recovery are as follows:

When the member joins the group, it connects with one of the seed members included in the list in its [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) system variable, initially using the [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) connection as specified in that list. The seed members might be a subset of the group.

Over this connection, the seed member uses Group Replication's membership service to provide the joining member with a list of all the members that are online in the group, in the form of a view. The membership information includes the details of the distributed recovery endpoints or standard SQL client connection offered by each member for distributed recovery.

The joining member selects a suitable group member from this list to be its donor for distributed recovery, following the behaviors described in [Section 18.5.3.4, “Fault Tolerance for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-fault).

The joining member then attempts to connect to the donor using the donor's advertised distributed recovery endpoints, trying each in turn in the order they are specified in the list. If the donor provides no endpoints, the joining member attempts to connect using the donor's standard SQL client connection. The SSL requirements for the connection are as specified by the **group\_replication\_recovery\_ssl\_\*** options described in [Section 18.5.3.1.4, “SSL and Authentication for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-connections-ssl).

If the joining member is not able to connect to the selected donor, it retries with other suitable donors, following the behaviors described in [Section 18.5.3.4, “Fault Tolerance for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-fault). Note that if the joining member exhausts the list of advertised endpoints without making a connection, it does not fall back to the donor's standard SQL client connection, but switches to another donor.

When the joining member establishes a distributed recovery connection with a donor, it uses that connection for state transfer as described in [Section 18.5.3, “Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery). The host and port for the connection that is used are shown in the joining member's log. Note that if a remote cloning operation is used, when the joining member has restarted at the end of the operation, it establishes a connection with a new donor for state transfer from the binary log. This might be a connection to a different member from the original donor used for the remote cloning operation, or it might be a different connection to the original donor. In any case, the distributed recovery process continues in the same way as it would have with the original donor.

##### 18.5.3.1.1 Selecting addresses for distributed recovery endpoints

IP addresses supplied by the [**group\_replication\_advertise\_recovery\_endpoints**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_advertise_recovery_endpoints) system variable as distributed recovery endpoints do not have to be configured for MySQL Server (that is, they do not have to be specified by the [**admin\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_admin_address) system variable or in the list for the [**bind\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_bind_address) system variable). They do have to be assigned to the server. Any host names used must resolve to a local IP address. IPv4 and IPv6 addresses can be used.

The ports supplied for the distributed recovery endpoints do have to be configured for MySQL Server, so they must be specified by the [**port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_port), [**report\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_port), or [**admin\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_admin_port) system variable. The server must listen for TCP/IP connections on these ports. If you specify the [**admin\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_admin_port), the replication user for distributed recovery needs the [**SERVICE\_CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_service-connection-admin) privilege to connect. Selecting the [**admin\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_admin_port) keeps distributed recovery connections separate from regular MySQL client connections.

Joining members try each of the endpoints in turn in the order they are specified on the list. If [**group\_replication\_advertise\_recovery\_endpoints**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_advertise_recovery_endpoints) is set to **DEFAULT** rather than a list of endpoints, the standard SQL client connection is offered. Note that the standard SQL client connection is not automatically included on a list of distributed recovery endpoints, and is not offered as a fallback if the donor's list of endpoints is exhausted without a connection. If you want to offer the standard SQL client connection as one of a number of distributed recovery endpoints, you must include it explicitly in the list specified by [**group\_replication\_advertise\_recovery\_endpoints**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_advertise_recovery_endpoints). You can put it in the last place so that it acts as a last resort for connection.

A group member's distributed recovery endpoints (or standard SQL client connection if endpoints are not provided) do not need to be added to the Group Replication allowlist specified by the [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) (from MySQL 8.0.22) or [**group\_replication\_ip\_whitelist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_whitelist) system variable. The allowlist is only for the address specified by [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) for each member. A joining member must have its initial connection to the group permitted by the allowlist in order to retrieve the address or addresses for distributed recovery.

The distributed recovery endpoints that you list are validated when the system variable is set and when a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement has been issued. If the list cannot be parsed correctly, or if any of the endpoints cannot be accessed on the host because the server is not listening on them, Group Replication logs an error and does not start.

##### 18.5.3.1.2 Compression for Distributed Recovery

From MySQL 8.0.18, you can optionally configure compression for distributed recovery by the method of state transfer from a donor's binary log. Compression can benefit distributed recovery where network bandwidth is limited and the donor has to transfer many transactions to the joining member. The [**group\_replication\_recovery\_compression\_algorithms**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_compression_algorithms) and [**group\_replication\_recovery\_zstd\_compression\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_zstd_compression_level) system variables configure permitted compression algorithms, and the **zstd** compression level, used when carrying out state transfer from a donor's binary log. For more information, see [Section 4.2.8, “Connection Compression Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#connection-compression-control).

Note that these compression settings do not apply for remote cloning operations. When a remote cloning operation is used for distributed recovery, the clone plugin's [**clone\_enable\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_enable_compression) setting applies.

##### 18.5.3.1.3 Replication User for Distributed Recovery

Distributed recovery requires a replication user that has the correct permissions so that Group Replication can establish direct member-to-member replication channels. The replication user must also have the correct permissions to act as the clone user on the donor for a remote cloning operation. The same replication user must be used for distributed recovery on every group member. For instructions to set up this replication user, see [Section 18.2.1.3, “User Credentials For Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-user-credentials). For instructions to secure the replication user credentials, see [Section 18.6.3.1, “Secure User Credentials for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-user).

##### 18.5.3.1.4 SSL and Authentication for Distributed Recovery

SSL for distributed recovery is configured separately from SSL for normal group communications, which is determined by the server's SSL settings and the [**group\_replication\_ssl\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ssl_mode) system variable. For distributed recovery connections, dedicated Group Replication distributed recovery SSL system variables are available to configure the use of certificates and ciphers specifically for distributed recovery.

By default, SSL is not used for distributed recovery connections. To activate it, set [**group\_replication\_recovery\_use\_ssl=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_use_ssl), and configure the Group Replication distributed recovery SSL system variables as described in [Section 18.6.3, “Securing Distributed Recovery Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-securing). You need a replication user that is set up to use SSL.

When distributed recovery is configured to use SSL, Group Replication applies this setting for remote cloning operations, as well as for state transfer from a donor's binary log. Group Replication automatically configures the settings for the clone SSL options ([**clone\_ssl\_ca**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_ssl_ca), [**clone\_ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_ssl_cert), and [**clone\_ssl\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_ssl_key)) to match your settings for the corresponding Group Replication distributed recovery options ([**group\_replication\_recovery\_ssl\_ca**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_ca), [**group\_replication\_recovery\_ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_cert), and [**group\_replication\_recovery\_ssl\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_key)).

If you are not using SSL for distributed recovery (so [**group\_replication\_recovery\_use\_ssl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_use_ssl) is set to **OFF**), and the replication user account for Group Replication authenticates with the **caching\_sha2\_password** plugin (which is the default in MySQL 8.0) or the **sha256\_password** plugin, RSA key-pairs are used for password exchange. In this case, either use the [**group\_replication\_recovery\_public\_key\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_public_key_path) system variable to specify the RSA public key file, or use the [**group\_replication\_recovery\_get\_public\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_get_public_key) system variable to request the public key from the source, as described in [Section 18.6.3.1.1, “Replication User With The Caching SHA-2 Authentication Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-caching-sha2-user-credentials).

#### 18.5.3.2 Cloning for Distributed Recovery

MySQL Server's clone plugin is available from MySQL 8.0.17. If you want to use remote cloning operations for distributed recovery in a group, you must set up existing members and joining members beforehand to support this function. If you do not want to use this function in a group, do not set it up, in which case Group Replication only uses state transfer from the binary log.

To use cloning, at least one existing group member and the joining member must be set up beforehand to support remote cloning operations. As a minimum, you must install the clone plugin on the donor and joining member, grant the [**BACKUP\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_backup-admin) permission to the replication user for distributed recovery, and set the [**group\_replication\_clone\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_clone_threshold) system variable to an appropriate level. To ensure the maximum availability of donors, it is advisable to set up all current and future group members to support remote cloning operations.

Be aware that a remote cloning operation removes user-created tablespaces and data from the joining member before transferring the data from the donor. If the operation is stopped while in progress, the joining member might be left with partial data or no data. This can be repaired by retrying the remote cloning operation, which Group Replication does automatically.

##### 18.5.3.2.1 Prerequisites for Cloning

For full instructions to set up and configure the clone plugin, see [Section 5.6.7, “The Clone Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin) . Detailed prerequisites for a remote cloning operation are covered in [Section 5.6.7.3, “Cloning Remote Data”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin-remote) . For Group Replication, note the following key points and differences:

The donor (an existing group member) and the recipient (the joining member) must have the clone plugin installed and active. For instructions to do this, see [Section 5.6.7.1, “Installing the Clone Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin-installation) .

The donor and the recipient must run on the same operating system, and must have the same MySQL Server version (which must be MySQL 8.0.17 or above to support the clone plugin). Cloning is therefore not suitable for groups where members run different MySQL Server versions.

The donor and the recipient must have the Group Replication plugin installed and active, and any other plugins that are active on the donor (such as a keyring plugin) must also be active on the recipient.

If distributed recovery is configured to use SSL ([**group\_replication\_recovery\_use\_ssl=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_use_ssl)), Group Replication applies this setting for remote cloning operations. Group Replication automatically configures the settings for the clone SSL options ([**clone\_ssl\_ca**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_ssl_ca), [**clone\_ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_ssl_cert), and [**clone\_ssl\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_ssl_key)) to match your settings for the corresponding Group Replication distributed recovery options ([**group\_replication\_recovery\_ssl\_ca**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_ca), [**group\_replication\_recovery\_ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_cert), and [**group\_replication\_recovery\_ssl\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_key)).

You do not need to set up a list of valid donors in the [**clone\_valid\_donor\_list**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_valid_donor_list) system variable for the purpose of joining a replication group. Group Replication configures this setting automatically for you after it selects a donor from the existing group members. Note that remote cloning operations use the server's SQL protocol hostname and port.

The clone plugin has a number of system variables to manage the network load and performance impact of the remote cloning operation. Group Replication does not configure these settings, so you can review them and set them if you want to, or allow them to default. Note that when a remote cloning operation is used for distributed recovery, the clone plugin's [**clone\_enable\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_enable_compression) setting applies to the operation, rather than the Group Replication compression setting.

To invoke the remote cloning operation on the recipient, Group Replication uses the internal **mysql.session** user, which already has the [**CLONE\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_clone-admin) privilege, so you do not need to set this up.

As the clone user on the donor for the remote cloning operation, Group Replication uses the replication user that you set up for distributed recovery (which is covered in [Section 18.2.1.3, “User Credentials For Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-user-credentials)). You must therefore give the [**BACKUP\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_backup-admin) privilege to this replication user on all group members that support cloning. Also give the privilege to the replication user on joining members when you are configuring them for Group Replication, because they can act as donors after they join the group. The same replication user is used for distributed recovery on every group member. To give this privilege to the replication user on existing members, you can issue this statement on each group member individually with binary logging disabled, or on one group member with binary logging enabled:

GRANT BACKUP\_ADMIN ON \*.\* TO ***rpl\_user***@'%';

If you use [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) to specify the replication user credentials on a server that previously supplied the user credentials using [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to), ensure that you remove the user credentials from the replication metadata repositories before any remote cloning operations take place. Also ensure that [**group\_replication\_start\_on\_boot=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) is set on the joining member. For instructions, see [Section 18.6.3, “Securing Distributed Recovery Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-securing). If you do not unset the user credentials, they are transferred to the joining member during remote cloning operations. The **group\_replication\_recovery** channel could then be inadvertently started with the stored credentials, on either the original member or members that were cloned from it. An automatic start of Group Replication on server boot (including after a remote cloning operation) would use the stored user credentials, and they would also be used if an operator did not specify the distributed recovery credentials on a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) command.

##### 18.5.3.2.2 Threshold for Cloning

When group members have been set up to support cloning, the [**group\_replication\_clone\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_clone_threshold) system variable specifies a threshold, expressed as a number of transactions, for the use of a remote cloning operation in distributed recovery. If the gap between the transactions on the donor and the transactions on the joining member is larger than this number, a remote cloning operation is used for state transfer to the joining member when this is technically possible. Group Replication calculates whether the threshold has been exceeded based on the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) sets of the existing group members. Using a remote cloning operation in the event of a large transaction gap lets you add new members to the group without transferring the group's data to the server manually beforehand, and also enables a member that is very out of date to catch up more efficiently.

The default setting for the [**group\_replication\_clone\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_clone_threshold) Group Replication system variable is extremely high (the maximum permitted sequence number for a transaction in a GTID), so it effectively deactivates cloning wherever state transfer from the binary log is possible. To enable Group Replication to select a remote cloning operation for state transfer where this is more appropriate, set the system variable to specify a number of transactions as the transaction gap above which you want cloning to take place.

**Warning**

Do not use a low setting for [**group\_replication\_clone\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_clone_threshold) in an active group. If a number of transactions above the threshold takes place in the group while the remote cloning operation is in progress, the joining member triggers a remote cloning operation again after restarting, and could continue this indefinitely. To avoid this situation, ensure that you set the threshold to a number higher than the number of transactions that you would expect to occur in the group during the time taken for the remote cloning operation.

Group Replication attempts to execute a remote cloning operation regardless of your threshold when state transfer from a donor's binary log is impossible, for example because the transactions needed by the joining member are not available in the binary log on any existing group member. Group Replication identifies this based on the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) sets of the existing group members. You cannot use the [**group\_replication\_clone\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_clone_threshold) system variable to deactivate cloning when the required transactions are not available in any member's binary log files, because in that situation cloning is the only alternative to transferring data to the joining member manually.

##### 18.5.3.2.3 Cloning Operations

When group members and joining members are set up for cloning, Group Replication manages remote cloning operations for you. A remote cloning operation might take some time to complete, depending on the size of the data. See [Section 5.6.7.9, “Monitoring Cloning Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin-monitoring) for information on monitoring the process.

**Note**

When state transfer is complete, Group Replication restarts the joining member to complete the process. If [**group\_replication\_start\_on\_boot=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) is set on the joining member, for example because you specify the replication user credentials on the [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement, you must issue [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) manually again following this restart. If [**group\_replication\_start\_on\_boot=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) and other settings required to start Group Replication were set in a configuration file or using a **SET PERSIST** statement, you do not need to intervene and the process continues automatically to bring the joining member online.

If the remote cloning procedure takes a long time, in releases before MySQL 8.0.22, it is possible for the set of certification information that accumulates for the group during that time to become too large to transmit to the joining member. In that case, the joining member logs an error message and does not join the group. From MySQL 8.0.22, Group Replication manages the garbage collection process for applied transactions differently to avoid this scenario. In earlier releases, if you do see this error, after the remote cloning operation completes, wait two minutes to allow a round of garbage collection to take place to reduce the size of the group's certification information. Then issue the following statement on the joining member, so that it stops trying to apply the previous set of certification information:

RESET SLAVE FOR CHANNEL group\_replication\_recovery;

Or from MySQL 8.0.22:

RESET REPLICA FOR CHANNEL group\_replication\_recovery;

A remote cloning operation clones settings that are persisted in tables from the donor to the recipient, as well as the data. Group Replication manages the settings that relate specifically to Group Replication channels. Group Replication member settings that are persisted in configuration files, such as the group replication local address, are not cloned and are not changed on the joining member. Group Replication also preserves the channel settings that relate to the use of SSL, so these are unique to the individual member.

If the replication user credentials used by the donor for the **group\_replication\_recovery** replication channel have been stored in the replication metadata repositories using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, they are transferred to and used by the joining member after cloning, and they must be valid there. With stored credentials, all group members that received state transfer by a remote cloning operation therefore automatically receive the replication user and password for distributed recovery. If you specify the replication user credentials on the [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement, these are used to start the remote cloning operation, but they are not transferred to and used by the joining member after cloning. If you do not want the credentials transferred to new joiners and recorded there, ensure that you unset them before remote cloning operations take place, as described in [Section 18.6.3, “Securing Distributed Recovery Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-securing), and use [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) to supply them instead.

If a **PRIVILEGE\_CHECKS\_USER** account has been used to help secure the replication appliers (see [Section 17.3.3.2, “Privilege Checks For Group Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks-gr)), from MySQL 8.0.19, the **PRIVILEGE\_CHECKS\_USER** account and related settings from the donor are cloned to the joining member. If the joining member is set to start Group Replication on boot, it automatically uses the account for privilege checks on the appropriate replication channels. (In MySQL 8.0.18, due to a number of limitations, it is recommended that you do not use a **PRIVILEGE\_CHECKS\_USER** account with Group Replication channels.)

##### 18.5.3.2.4 Cloning for Other Purposes

Group Replication initiates and manages cloning operations for distributed recovery. Group members that have been set up to support cloning may also participate in cloning operations that a user initiates manually. For example, you might want to create a new server instance by cloning from a group member as the donor, but you do not want the new server instance to join the group immediately, or maybe not ever.

In all releases that support cloning, you can initiate a cloning operation manually involving a group member on which Group Replication is stopped. Note that because cloning requires that the active plugins on a donor and recipient must match, the Group Replication plugin must be installed and active on the other server instance, even if you do not intend that server instance to join a group. You can install the plugin by issuing this statement:

INSTALL PLUGIN group\_replication SONAME 'group\_replication.so';

In releases before MySQL 8.0.20, you cannot initiate a cloning operation manually if the operation involves a group member on which Group Replication is running. From MySQL 8.0.20, you can do this, provided that the cloning operation does not remove and replace the data on the recipient. The statement to initiate the cloning operation must therefore include the **DATA DIRECTORY** clause if Group Replication is running.

#### 18.5.3.3 Configuring Distributed Recovery

Several aspects of Group Replication's distributed recovery process can be configured to suit your system.

##### Number of Connection Attempts

For state transfer from the binary log, Group Replication limits the number of attempts a joining member makes when trying to connect to a donor from the pool of donors. If the connection retry limit is reached without a successful connection, the distributed recovery procedure terminates with an error. Note that this limit specifies the total number of attempts that the joining member makes to connect to a donor. For example, if 2 group members are suitable donors, and the connection retry limit is set to 4, the joining member makes 2 attempts to connect to each of the donors before reaching the limit.

The default connection retry limit is 10. You can configure this setting using the [**group\_replication\_recovery\_retry\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_retry_count) system variable. The following command sets the maximum number of attempts to connect to a donor to 5:

mysql> **SET GLOBAL group\_replication\_recovery\_retry\_count= 5;**

For remote cloning operations, this limit does not apply. Group Replication makes only one connection attempt to each suitable donor for cloning, before starting to attempt state transfer from the binary log.

##### Sleep Interval for Connection Attempts

For state transfer from the binary log, the [**group\_replication\_recovery\_reconnect\_interval**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_reconnect_interval) system variable defines how much time the distributed recovery process should sleep between donor connection attempts. Note that distributed recovery does not sleep after every donor connection attempt. As the joining member is connecting to different servers and not to the same one repeatedly, it can assume that the problem that affects server A does not affect server B. Distributed recovery therefore suspends only when it has gone through all the possible donors. Once the server joining the group has made one attempt to connect to each of the suitable donors in the group, the distributed recovery process sleeps for the number of seconds configured by the [**group\_replication\_recovery\_reconnect\_interval**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_reconnect_interval) system variable. For example, if 2 group members are suitable donors, and the connection retry limit is set to 4, the joining member makes one attempt to connect to each of the donors, then sleeps for the connection retry interval, then makes one further attempt to connect to each of the donors before reaching the limit.

The default connection retry interval is 60 seconds, and you can change this value dynamically. The following command sets the distributed recovery donor connection retry interval to 120 seconds:

mysql> **SET GLOBAL group\_replication\_recovery\_reconnect\_interval= 120;**

For remote cloning operations, this interval does not apply. Group Replication makes only one connection attempt to each suitable donor for cloning, before starting to attempt state transfer from the binary log.

##### Marking the Joining Member Online

When distributed recovery has successfully completed state transfer from the donor to the joining member, the joining member can be marked as online in the group and ready to participate. By default, this is done after the joining member has received and applied all the transactions that it was missing. Optionally, you can allow a joining member to be marked as online when it has received and certified (that is, completed conflict detection for) all the transactions that it was missing, but before it has applied them. If you want to do this, use the [**group\_replication\_recovery\_complete\_at**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_complete_at) system variable to specify the alternative setting **TRANSACTIONS\_CERTIFIED**.

#### 18.5.3.4 Fault Tolerance for Distributed Recovery

Group Replication's distributed recovery process has a number of built-in measures to ensure fault tolerance in the event of any problems during the process.

The donor for distributed recovery is selected randomly from the existing list of suitable online group members in the current view. Selecting a random donor means that there is a good chance that the same server is not selected more than once when multiple members enter the group. From MySQL 8.0.17, for state transfer from the binary log, the joiner only selects a donor that is running a lower or equal patch version of MySQL Server compared to itself. For earlier releases, all of the online members are allowed to be a donor. For a remote cloning operation, the joiner only selects a donor that is running the same patch version as itself. Note that when the joining member has restarted at the end of the operation, it establishes a connection with a new donor for state transfer from the binary log, which might be a different member from the original donor used for the remote cloning operation.

In the following situations, Group Replication detects an error in distributed recovery, automatically switches over to a new donor, and retries the state transfer:

Connection error - There is an authentication issue or another problem with making the connection to a candidate donor.

Replication errors - One of the replication threads (the receiver or applier threads) being used for state transfer from the binary log fails. Because this method of state transfer uses the existing MySQL replication framework, it is possible that some transient errors could cause errors in the receiver or applier threads.

Remote cloning operation errors - A remote cloning operation fails or is stopped before it completes.

Donor leaves the group - The donor leaves the group, or Group Replication is stopped on the donor, while state transfer is in progress.

The Performance Schema table [**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table) displays the error that caused the last retry. In these situations, the new connection following the error is attempted with a new candidate donor. Selecting a different donor in the event of an error means that there is a chance the new candidate donor does not have the same error. If the clone plugin is installed, Group Replication attempts a remote cloning operation with each of the suitable online clone-supporting donors first. If all those attempts fail, Group Replication attempts state transfer from the binary log with all the suitable donors in turn, if that is possible.

**Warning**

For a remote cloning operation, user-created tablespaces and data on the recipient (the joining member) are dropped before the remote cloning operation begins to transfer the data from the donor. If the remote cloning operation starts but does not complete, the joining member might be left with a partial set of its original data files, or with no user data. Data transferred by the donor is removed from the recipient if the cloning operation is stopped before the data is fully cloned. This situation can be repaired by retrying the cloning operation, which Group Replication does automatically.

In the following situations, the distributed recovery process cannot be completed, and the joining member leaves the group:

Purged transactions - Transactions that are required by the joining member are not present in any online group member's binary log files, and the data cannot be obtained by a remote cloning operation (because the clone plugin is not installed, or because cloning was attempted with all possible donors but failed). The joining member is therefore unable to catch up with the group.

Conflicting transactions - The joining member already contains some transactions that are not present in the group. If a remote cloning operation was carried out, these transactions would be deleted and lost, because the data directory on the joining member is erased. If state transfer from a donor's binary log was carried out, these transactions could conflict with the group's transactions.

Connection retry limit reached - The joining member has made all the connection attempts allowed by the connection retry limit. You can configure this using the [**group\_replication\_recovery\_retry\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_retry_count) system variable (see [Section 18.5.3.3, “Configuring Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-tuning-recovery)).

No more donors - The joining member has unsuccessfully attempted a remote cloning operation with each of the online clone-supporting donors in turn (if the clone plugin is installed), then has unsuccessfully attempted state transfer from the binary log with each of the suitable online donors in turn, if possible.

Joining member leaves the group - The joining member leaves the group or Group Replication is stopped on the joining member while state transfer is in progress.

If the joining member left the group unintentionally, so in any situation listed above except the last, it proceeds to take the action specified by the [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action) system variable.

#### 18.5.3.5 How Distributed Recovery Works

When Group Replication's distributed recovery process is carrying out state transfer from the binary log, to synchronize the joining member with the donor up to a specific point in time, the joining member and donor make use of GTIDs (see [Section 17.1.3, “Replication with Global Transaction Identifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids)). However, GTIDs only provide a means to realize which transactions the joining member is missing. They do not help marking a specific point in time to which the server joining the group must catch up, nor do they convey certification information. This is the job of binary log view markers, which mark view changes in the binary log stream, and also contain additional metadata information, supplying the joining member with missing certification-related data.

This topic explains the role of view changes and the view change identifier, and the steps to carry out state transfer from the binary log.

##### View and View Changes

A view corresponds to a group of members participating actively in the current configuration, in other words at a specific point in time. They are functioning correctly and online in the group.

A view change occurs when a modification to the group configuration happens, such as a member joining or leaving. Any group membership change results in an independent view change communicated to all members at the same logical point in time.

A view identifier uniquely identifies a view. It is generated whenever a view change happens.

At the group communication layer, view changes with their associated view identifiers mark boundaries between the data exchanged before and after a member joins. This concept is implemented through a binary log event: the"view change log event". The view identifier is recorded to demarcate transactions transmitted before and after changes happen in the group membership.

The view identifier itself is built from two parts: a randomly generated part, and a monotonically increasing integer. The randomly generated part is generated when the group is created, and remains unchanged while there is at least one member in the group. The integer is incremented every time a view change happens. Using these two different parts enables the view identifier to identify incremental group changes caused by members joining or leaving, and also to identify the situation where all members leave the group in a full group shutdown, so no information remains of what view the group was in. Randomly generating part of the identifier when the group is started from the beginning ensures that the data markers in the binary log remain unique, and an identical identifier is not reused after a full group shutdown, as this would cause issues with distributed recovery in the future.

##### Begin: Stable Group

All servers are online and processing incoming transactions from the group. Some servers may be a little behind in terms of transactions replicated, but eventually they converge. The group acts as one distributed and replicated database.

**Figure 18.8 Stable Group**

##### View Change: a Member Joins

Whenever a new member joins the group and therefore a view change is performed, every online server queues a view change log event for execution. This is queued because before the view change, several transactions can be queued on the server to be applied and as such, these belong to the old view. Queuing the view change event after them guarantees a correct marking of when this happened.

Meanwhile, the joining member selects a suitable donor the donor from the list of online servers as stated by the membership service through the view abstraction. A member joins on view 4 and the online members write a view change event to the binary log.

**Figure 18.9 A Member Joins**

##### State Transfer: Catching Up

If group members and the joining member are set up with the clone plugin (see [Section 18.5.3.2, “Cloning for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-cloning)), and the difference in transactions between the joining member and the group exceeds the threshold set for a remote cloning operation ([**group\_replication\_clone\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_clone_threshold)), Group Replication begins distributed recovery with a remote cloning operation. A remote cloning operation is also carried out if required transactions are no longer present in any group member's binary log files. During a remote cloning operation, the existing data on the joining member is removed, and replaced with a copy of the donor's data. When the remote cloning operation is complete and the joining member has restarted, state transfer from a donor's binary log is carried out to get the transactions that the group applied while the remote cloning operation was in progress. If there is not a large transaction gap, or if the clone plugin is not installed, Group Replication proceeds directly to state transfer from a donor's binary log.

For state transfer from a donor's binary log, a connection is established between the joining member and the donor and state transfer begins. This interaction with the donor continues until the server joining the group's applier thread processes the view change log event that corresponds to the view change triggered when the server joining the group came into the group. In other words, the server joining the group replicates from the donor, until it gets to the marker with the view identifier which matches the view marker it is already in.

**Figure 18.10 State Transfer: Catching Up**

As view identifiers are transmitted to all members in the group at the same logical time, the server joining the group knows at which view identifier it should stop replicating. This avoids complex GTID set calculations because the view identifier clearly marks which data belongs to each group view.

While the server joining the group is replicating from the donor, it is also caching incoming transactions from the group. Eventually, it stops replicating from the donor and switches to applying those that are cached.

**Figure 18.11 Queued Transactions**

##### Finish: Caught Up

When the server joining the group recognizes a view change log event with the expected view identifier, the connection to the donor is terminated and it starts applying the cached transactions. Although it acts as a marker in the binary log, delimiting view changes, the view change log event also plays another role. It conveys the certification information as perceived by all servers when the server joining the group entered the group, in other words the last view change. Without it, the server joining the group would not have the necessary information to be able to certify (detect conflicts) subsequent transactions.

The duration of the catch up is not deterministic, because it depends on the workload and the rate of incoming transactions to the group. This process is completely online and the server joining the group does not block any other server in the group while it is catching up. Therefore the number of transactions the server joining the group is behind when it moves to this stage can, for this reason, vary and thus increase or decrease according to the workload.

When the server joining the group reaches zero queued transactions and its stored data is equal to the other members, its public state changes to online.

**Figure 18.12 Instance Online**

### 18.5.4 Network Partitioning

The group needs to achieve consensus whenever a change that needs to be replicated happens. This is the case for regular transactions but is also required for group membership changes and some internal messaging that keeps the group consistent. Consensus requires a majority of group members to agree on a given decision. When a majority of group members is lost, the group is unable to progress and blocks because it cannot secure majority or quorum.

Quorum may be lost when there are multiple involuntary failures, causing a majority of servers to be removed abruptly from the group. For example in a group of 5 servers, if 3 of them become silent at once, the majority is compromised and thus no quorum can be achieved. In fact, the remaining two are not able to tell if the other 3 servers have crashed or whether a network partition has isolated these 2 alone and therefore the group cannot be reconfigured automatically.

On the other hand, if servers exit the group voluntarily, they instruct the group that it should reconfigure itself. In practice, this means that a server that is leaving tells others that it is going away. This means that other members can reconfigure the group properly, the consistency of the membership is maintained and the majority is recalculated. For example, in the above scenario of 5 servers where 3 leave at once, if the 3 leaving servers warn the group that they are leaving, one by one, then the membership is able to adjust itself from 5 to 2, and at the same time, securing quorum while that happens.

**Note**

Loss of quorum is by itself a side-effect of bad planning. Plan the group size for the number of expected failures (regardless whether they are consecutive, happen all at once or are sporadic).

The following sections explain what to do if the system partitions in such a way that no quorum is automatically achieved by the servers in the group.

Tip

A primary that has been excluded from a group after a majority loss followed by a reconfiguration can contain extra transactions that are not included in the new group. If this happens, the attempt to add back the excluded member from the group results in an error with the message This member has more executed transactions than those present in the group.

#### Detecting Partitions

The [**replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table) performance schema table presents the status of each server in the current view from the perspective of this server. The majority of the time the system does not run into partitioning, and therefore the table shows information that is consistent across all servers in the group. In other words, the status of each server on this table is agreed by all in the current view. However, if there is network partitioning, and quorum is lost, then the table shows the status **UNREACHABLE** for those servers that it cannot contact. This information is exported by the local failure detector built into Group Replication.

**Figure 18.13 Losing Quorum**

To understand this type of network partition the following section describes a scenario where there are initially 5 servers working together correctly, and the changes that then happen to the group once only 2 servers are online. The scenario is depicted in the figure.

As such, lets assume that there is a group with these 5 servers in it:

Server s1 with member identifier **199b2df7-4aaf-11e6-bb16-28b2bd168d07**

Server s2 with member identifier **199bb88e-4aaf-11e6-babe-28b2bd168d07**

Server s3 with member identifier **1999b9fb-4aaf-11e6-bb54-28b2bd168d07**

Server s4 with member identifier **19ab72fc-4aaf-11e6-bb51-28b2bd168d07**

Server s5 with member identifier **19b33846-4aaf-11e6-ba81-28b2bd168d07**

Initially the group is running fine and the servers are happily communicating with each other. You can verify this by logging into s1 and looking at its [**replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table) performance schema table. For example:

mysql> **SELECT MEMBER\_ID,MEMBER\_STATE, MEMBER\_ROLE FROM performance\_schema.replication\_group\_members;**

+--------------------------------------+--------------+-------------+

| MEMBER\_ID | MEMBER\_STATE |-MEMBER\_ROLE |

+--------------------------------------+--------------+-------------+

| 1999b9fb-4aaf-11e6-bb54-28b2bd168d07 | ONLINE | SECONDARY |

| 199b2df7-4aaf-11e6-bb16-28b2bd168d07 | ONLINE | PRIMARY |

| 199bb88e-4aaf-11e6-babe-28b2bd168d07 | ONLINE | SECONDARY |

| 19ab72fc-4aaf-11e6-bb51-28b2bd168d07 | ONLINE | SECONDARY |

| 19b33846-4aaf-11e6-ba81-28b2bd168d07 | ONLINE | SECONDARY |

+--------------------------------------+--------------+-------------+

However, moments later there is a catastrophic failure and servers s3, s4 and s5 stop unexpectedly. A few seconds after this, looking again at the [**replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table) table on s1 shows that it is still online, but several others members are not. In fact, as seen below they are marked as **UNREACHABLE**. Moreover, the system could not reconfigure itself to change the membership, because the majority has been lost.

mysql> **SELECT MEMBER\_ID,MEMBER\_STATE FROM performance\_schema.replication\_group\_members;**

+--------------------------------------+--------------+

| MEMBER\_ID | MEMBER\_STATE |

+--------------------------------------+--------------+

| 1999b9fb-4aaf-11e6-bb54-28b2bd168d07 | UNREACHABLE |

| 199b2df7-4aaf-11e6-bb16-28b2bd168d07 | ONLINE |

| 199bb88e-4aaf-11e6-babe-28b2bd168d07 | ONLINE |

| 19ab72fc-4aaf-11e6-bb51-28b2bd168d07 | UNREACHABLE |

| 19b33846-4aaf-11e6-ba81-28b2bd168d07 | UNREACHABLE |

+--------------------------------------+--------------+

The table shows that s1 is now in a group that has no means of progressing without external intervention, because a majority of the servers are unreachable. In this particular case, the group membership list needs to be reset to allow the system to proceed, which is explained in this section. Alternatively, you could also choose to stop Group Replication on s1 and s2 (or stop completely s1 and s2), figure out what happened with s3, s4 and s5 and then restart Group Replication (or the servers).

#### Unblocking a Partition

Group replication enables you to reset the group membership list by forcing a specific configuration. For instance in the case above, where s1 and s2 are the only servers online, you could choose to force a membership configuration consisting of only s1 and s2. This requires checking some information about s1 and s2 and then using the [**group\_replication\_force\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_force_members) variable.

**Figure 18.14 Forcing a New Membership**

Suppose that you are back in the situation where s1 and s2 are the only servers left in the group. Servers s3, s4 and s5 have left the group unexpectedly. To make servers s1 and s2 continue, you want to force a membership configuration that contains only s1 and s2.

**Warning**

This procedure uses [**group\_replication\_force\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_force_members) and should be considered a last resort remedy. It must be used with extreme care and only for overriding loss of quorum. If misused, it could create an artificial split-brain scenario or block the entire system altogether.

Recall that the system is blocked and the current configuration is the following (as perceived by the local failure detector on s1):

mysql> **SELECT MEMBER\_ID,MEMBER\_STATE FROM performance\_schema.replication\_group\_members;**

+--------------------------------------+--------------+

| MEMBER\_ID | MEMBER\_STATE |

+--------------------------------------+--------------+

| 1999b9fb-4aaf-11e6-bb54-28b2bd168d07 | UNREACHABLE |

| 199b2df7-4aaf-11e6-bb16-28b2bd168d07 | ONLINE |

| 199bb88e-4aaf-11e6-babe-28b2bd168d07 | ONLINE |

| 19ab72fc-4aaf-11e6-bb51-28b2bd168d07 | UNREACHABLE |

| 19b33846-4aaf-11e6-ba81-28b2bd168d07 | UNREACHABLE |

+--------------------------------------+--------------+

The first thing to do is to check what is the local address (group communication identifier) for s1 and s2. Log in to s1 and s2 and get that information as follows.

mysql> **SELECT @@group\_replication\_local\_address;**

Once you know the group communication addresses of s1 (**127.0.0.1:10000**) and s2 (**127.0.0.1:10001**), you can use that on one of the two servers to inject a new membership configuration, thus overriding the existing one that has lost quorum. To do that on s1:

mysql> **SET GLOBAL group\_replication\_force\_members="**127.0.0.1:10000,127.0.0.1:10001";

This unblocks the group by forcing a different configuration. Check [**replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table) on both s1 and s2 to verify the group membership after this change. First on s1.

mysql> **SELECT MEMBER\_ID,MEMBER\_STATE FROM performance\_schema.replication\_group\_members;**

+--------------------------------------+--------------+

| MEMBER\_ID | MEMBER\_STATE |

+--------------------------------------+--------------+

| b5ffe505-4ab6-11e6-b04b-28b2bd168d07 | ONLINE |

| b60907e7-4ab6-11e6-afb7-28b2bd168d07 | ONLINE |

+--------------------------------------+--------------+

And then on s2.

mysql> **SELECT \* FROM performance\_schema.replication\_group\_members;**

+--------------------------------------+--------------+

| MEMBER\_ID | MEMBER\_STATE |

+--------------------------------------+--------------+

| b5ffe505-4ab6-11e6-b04b-28b2bd168d07 | ONLINE |

| b60907e7-4ab6-11e6-afb7-28b2bd168d07 | ONLINE |

+--------------------------------------+--------------+

When forcing a new membership configuration, make sure that any servers are going to be forced out of the group are indeed stopped. In the scenario depicted above, if s3, s4 and s5 are not really unreachable but instead are online, they may have formed their own functional partition (they are 3 out of 5, hence they have the majority). In that case, forcing a group membership list with s1 and s2 could create an artificial split-brain situation. Therefore it is important before forcing a new membership configuration to ensure that the servers to be excluded are indeed shut down and if they are not, shut them down before proceeding.

After you have used the [**group\_replication\_force\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_force_members) system variable to successfully force a new group membership and unblock the group, ensure that you clear the system variable. [**group\_replication\_force\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_force_members) must be empty in order to issue a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement.

### 18.5.5 Support For IPv6 And For Mixed IPv6 And IPv4 Groups

From MySQL 8.0.14, Group Replication group members can use IPv6 addresses as an alternative to IPv4 addresses for communications within the group. To use IPv6 addresses, the operating system on the server host and the MySQL Server instance must both be configured to support IPv6. For instructions to set up IPv6 support for a server instance, see [Section 5.1.13, “IPv6 Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#ipv6-support).

IPv6 addresses, or host names that resolve to them, can be specified as the network address that the member provides in the [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) option for connections from other members. When specified with a port number, an IPv6 address must be specified in square brackets, for example:

group\_replication\_local\_address= "[2001:db8:85a3:8d3:1319:8a2e:370:7348]:33061"

The network address or host name specified in [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) is used by Group Replication as the unique identifier for a group member within the replication group. If a host name specified as the Group Replication local address for a server instance resolves to both an IPv4 and an IPv6 address, the IPv4 address is always used for Group Replication connections. The address or host name specified as the Group Replication local address is not the same as the MySQL server SQL protocol host and port, and is not specified in the [**bind\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_bind_address) system variable for the server instance. For the purpose of IP address permissions for Group Replication (see [Section 18.6.1, “Group Replication IP Address Permissions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-ip-address-permissions)), the address that you specify for each group member in [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) must be added to the list for the [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) (from MySQL 8.0.22) or [**group\_replication\_ip\_whitelist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_whitelist) system variable on the other servers in the replication group.

A replication group can contain a combination of members that present an IPv6 address as their Group Replication local address, and members that present an IPv4 address. When a server joins such a mixed group, it must make the initial contact with the seed member using the protocol that the seed member advertises in the [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) option, whether that is IPv4 or IPv6. If any of the seed members for the group are listed in the [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) option with an IPv6 address when a joining member has an IPv4 Group Replication local address, or the reverse, you must also set up and permit an alternative address for the joining member for the required protocol (or a host name that resolves to an address for that protocol). If a joining member does not have a permitted address for the appropriate protocol, its connection attempt is refused. The alternative address or host name only needs to be added to the [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) (from MySQL 8.0.22) or [**group\_replication\_ip\_whitelist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_whitelist) system variable on the other servers in the replication group, not to the [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) value for the joining member (which can only contain a single address).

For example, server A is a seed member for a group, and has the following configuration settings for Group Replication, so that it is advertising an IPv6 address in the [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) option:

group\_replication\_bootstrap\_group=on

group\_replication\_local\_address= "[2001:db8:85a3:8d3:1319:8a2e:370:7348]:33061"

group\_replication\_group\_seeds= "[2001:db8:85a3:8d3:1319:8a2e:370:7348]:33061"

Server B is a joining member for the group, and has the following configuration settings for Group Replication, so that it has an IPv4 Group Replication local address:

group\_replication\_bootstrap\_group=off

group\_replication\_local\_address= "203.0.113.21:33061"

group\_replication\_group\_seeds= "[2001:db8:85a3:8d3:1319:8a2e:370:7348]:33061"

Server B also has an alternative IPv6 address **2001:db8:8b0:40:3d9c:cc43:e006:19e8**. For Server B to join the group successfully, both its IPv4 Group Replication local address, and its alternative IPv6 address, must be listed in Server A's allowlist, as in the following example:

group\_replication\_ip\_allowlist=

"203.0.113.0/24,2001:db8:85a3:8d3:1319:8a2e:370:7348,

2001:db8:8b0:40:3d9c:cc43:e006:19e8"

As a best practice for Group Replication IP address permissions, Server B (and all other group members) should have the same allowlist as Server A, unless security requirements demand otherwise.

If any or all members of a replication group are using an older MySQL Server version that does not support the use of IPv6 addresses for Group Replication, a member cannot participate in the group using an IPv6 address (or a host name that resolves to one) as its Group Replication local address. This applies both in the case where at least one existing member uses an IPv6 address and a new member that does not support this attempts to join, and in the case where a new member attempts to join using an IPv6 address but the group includes at least one member that does not support this. In each situation, the new member cannot join. To make a joining member present an IPv4 address for group communications, you can either change the value of [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) to an IPv4 address, or configure your DNS to resolve the joining member's existing host name to an IPv4 address. After you have upgraded every group member to a MySQL Server version that supports IPv6 for Group Replication, you can change the [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) value for each member to an IPv6 address, or configure your DNS to present an IPv6 address. Changing the value of [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) takes effect only when you stop and restart Group Replication.

IPv6 addresses can also be used as distributed recovery endpoints, which can be specified from MySQL 8.0.21 using the [**group\_replication\_advertise\_recovery\_endpoints**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_advertise_recovery_endpoints) system variable. The same rules apply to addresses used in this list. See [Section 18.5.3.1, “Connections for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-connections).

### 18.5.6 Using MySQL Enterprise Backup with Group Replication

[MySQL Enterprise Backup](https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/" \t "_top) is a commercially-licensed backup utility for MySQL Server, available with [MySQL Enterprise Edition](https://www.mysql.com/products/enterprise/). This section explains how to back up and subsequently restore a Group Replication member using MySQL Enterprise Backup. The same technique can be used to quickly add a new member to a group.

#### Backing up a Group Replication Member Using MySQL Enterprise Backup

Backing up a Group Replication member is similar to backing up a stand-alone MySQL instance. The following instructions assume that you are already familiar with how to use MySQL Enterprise Backup to perform a backup; if that is not the case, please review the [MySQL Enterprise Backup 8.0 User's Guide](https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/), especially [Backing Up a Database Server](https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/backing-up.html). Also note the requirements described in [Grant MySQL Privileges to Backup Administrator](https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/mysqlbackup.privileges.html) and [Using MySQL Enterprise Backup with Group Replication](https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/meb-group-replication.html).

Consider the following group with three members, **s1**, **s2**, and **s3**, running on hosts with the same names:

mysql> **SELECT member\_host, member\_port, member\_state FROM performance\_schema.replication\_group\_members;**

+-------------+-------------+--------------+

| member\_host | member\_port | member\_state |

+-------------+-------------+--------------+

| s1 | 3306 | ONLINE |

| s2 | 3306 | ONLINE |

| s3 | 3306 | ONLINE |

+-------------+-------------+--------------+

Using MySQL Enterprise Backup, create a backup of **s2** by issuing on its host, for example, the following command:

s2> **mysqlbackup --defaults-file=/etc/my.cnf --backup-image=/backups/my.mbi\_`date +%d%m\_%H%M` \**

**--backup-dir=/backups/backup\_`date +%d%m\_%H%M` --user=root -p \**

**--host=127.0.0.1 backup-to-image**

**Notes**

For MySQL Enterprise Backup 8.0.18 and earlier, If the system variable [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) is set to **ON** for the group, MySQL Enterprise Backup is not able to log the backup progress on the servers. This is because the **backup\_progress** table on the server is a CSV table, for which primary keys are not supported. In that case, **mysqlbackup** issues the following warnings during the backup operation:

181011 11:17:06 MAIN WARNING: MySQL query 'CREATE TABLE IF NOT EXISTS

mysql.backup\_progress( `backup\_id` BIGINT NOT NULL, `tool\_name` VARCHAR(4096)

NOT NULL, `error\_code` INT NOT NULL, `error\_message` VARCHAR(4096) NOT NULL,

`current\_time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ON

UPDATE CURRENT\_TIMESTAMP,`current\_state` VARCHAR(200) NOT NULL ) ENGINE=CSV

DEFAULT CHARSET=utf8 COLLATE=utf8\_bin': 3750, Unable to create a table

without PK, when system variable 'sql\_require\_primary\_key' is set. Add a PK

to the table or unset this variable to avoid this message. Note that tables

without PK can cause performance problems in row-based replication, so please

consult your DBA before changing this setting.

181011 11:17:06 MAIN WARNING: This backup operation's progress info cannot be

logged.

This does not prevent **mysqlbackup** from finishing the backup though.

For MySQL Enterprise Backup 8.0.20 and earlier, when backing up a secondary member, as MySQL Enterprise Backup cannot write backup status and metadata to a read-only server instance, it might issue warnings similar to the following one during the backup operation:

181113 21:31:08 MAIN WARNING: This backup operation cannot write to backup

progress. The MySQL server is running with the --super-read-only option.

You can avoid the warning by using the --no-history-logging option with your backup command. This is not an issue for MySQL Enterprise Backup 8.0.21 and higher—see [Using MySQL Enterprise Backup with Group Replication](https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/meb-group-replication.html) for details.

#### Restoring a Failed Member

Assume one of the members (**s3** in the following example) is irreconcilably corrupted. The most recent backup of group member **s2** can be used to restore **s3**. Here are the steps for performing the restore:

Copy the backup of s2 onto the host for s3. The exact way to copy the backup depends on the operating system and tools available to you. In this example, we assume the hosts are both Linux servers and use SCP to copy the files between them:

s2/backups> **scp my.mbi\_2206\_1429 s3:/backups**

Restore the backup. Connect to the target host (the host for **s3** in this case), and restore the backup using MySQL Enterprise Backup. Here are the steps:

Stop the corrupted server, if it is still running. For example, on Linux distributions that use systemd:

s3> systemctl stop mysqld

Preserve the two configuration files in the corrupted server's data directory, auto.cnf and mysqld-auto.cnf (if it exists), by copying them to a safe location outside of the data directory. This is for preserving the [server's UUID](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) and [Section 5.1.9.3, “Persisted System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#persisted-system-variables) (if used), which are needed in the steps below.

Delete all contents in the data directory of **s3**. For example:

s3> rm -rf /var/lib/mysql/\*

If the system variables [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), [**innodb\_log\_group\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_group_home_dir), and [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) point to any directories other than the data directory, they should also be made empty; otherwise, the restore operation fails.

Restore backup of **s2** onto the host for **s3**:

s3> **mysqlbackup --defaults-file=/etc/my.cnf \**

**--datadir=/var/lib/mysql \**

**--backup-image=/backups/my.mbi\_2206\_1429 \**

**--backup-dir=/tmp/restore\_`date +%d%m\_%H%M` copy-back-and-apply-log**

**Note**

The command above assumes that the binary logs and relay logs on **s2** and **s3** have the same base name and are at the same location on the two servers. If these conditions are not met, you should use the [--log-bin](https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/server-repository-options.html#option_meb_log-bin) and [--relay-log](https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/server-repository-options.html#option_meb_relay-log) options to restore the binary log and relay log to their original file paths on **s3**. For example, if you know that on **s3** the binary log's base name is **s3-bin** and the relay-log's base name is **s3-relay-bin**, your restore command should look like:

**mysqlbackup --defaults-file=/etc/my.cnf \**

**--datadir=/var/lib/mysql \**

**--backup-image=/backups/my.mbi\_2206\_1429 \**

**--log-bin=s3-bin --relay-log=s3-relay-bin \**

**--backup-dir=/tmp/restore\_`date +%d%m\_%H%M` copy-back-and-apply-log**

Being able to restore the binary log and relay log to the right file paths makes the restore process easier; if that is impossible for some reason, see [Rebuild the Failed Member to Rejoin as a New Member](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-rebuild-member).

Restore the *auto.cnf* file for s3. To rejoin the replication group, the restored member must have the same [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) it used to join the group before. Supply the old server UUID by copying the auto.cnf file preserved in step 2 above into the data directory of the restored member.

**Note**

If you cannot supply the failed member's original [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) to the restored member by restoring its old auto.cnf file, you must let the restored member join the group as a new member; see instructions in [Rebuild the Failed Member to Rejoin as a New Member](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-rebuild-member) below on how to do that.

Restore the *mysqld-auto.cnf* file for s3 (only required if s3 used persistent system variables). The settings for the [Section 5.1.9.3, “Persisted System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#persisted-system-variables) that were used to configure the failed member must be provided to the restored member. These settings are to be found in the mysqld-auto.cnf file of the failed server, which you should have preserved in step 2 above. Restore the file to the data directory of the restored server. See [Restoring Persisted System Variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-meb-restore-persisted-variables) on what to do if you do not have a copy of the file.

Start the restored server. For example, on Linux distributions that use systemd:

systemctl start mysqld

**Note**

If the server you are restoring is a primary member, perform the steps described in [Restoring a Primary Member](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-meb-restore-primary) before starting the restored server.

Restart Group Replication. Connect to the restarted **s3** using, for example, a [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client, and issue the following command:

mysql> START GROUP\_REPLICATION;

Before the restored instance can become an online member of the group, it needs to apply any transactions that have happened to the group after the backup was taken; this is achieved using Group Replication's [distributed recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery) mechanism, and the process starts after the [START GROUP\_REPLICATION](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement has been issued. To check the member status of the restored instance, issue:

mysql> SELECT member\_host, member\_port, member\_state FROM performance\_schema.replication\_group\_members;

+-------------+-------------+--------------+

| member\_host | member\_port | member\_state |

+-------------+-------------+--------------+

| s1 | 3306 | ONLINE |

| s2 | 3306 | ONLINE |

| s3 | 3306 | RECOVERING |

+-------------+-------------+--------------+

This shows that **s3** is applying transactions to catch up with the group. Once it has caught up with the rest of the group, its **member\_state** changes to **ONLINE**:

mysql> SELECT member\_host, member\_port, member\_state FROM performance\_schema.replication\_group\_members;

+-------------+-------------+--------------+

| member\_host | member\_port | member\_state |

+-------------+-------------+--------------+

| s1 | 3306 | ONLINE |

| s2 | 3306 | ONLINE |

| s3 | 3306 | ONLINE |

+-------------+-------------+--------------+

**Note**

If the server you are restoring is a primary member, once it has gained synchrony with the group and become **ONLINE**, perform the steps described at the end of [Restoring a Primary Member](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-meb-restore-primary) to revert the configuration changes you had made to the server before you started it.

The member has now been fully restored from the backup and functions as a regular member of the group.

#### Rebuild the Failed Member to Rejoin as a New Member

Sometimes, the steps outlined above in [Restoring a Failed Member](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-restore-failed-member) cannot be carried out because, for example, the binary log or relay log is corrupted, or it is just missing from the backup. In such a situation, use the backup to rebuild the member, and then add it to the group as a new member. In the steps below, we assume the rebuilt member is named **s3**, like the failed member, and that it runs on the same host as **s3**:

Copy the backup of s2 onto the host for s3 . The exact way to copy the backup depends on the operating system and tools available to you. In this example we assume the hosts are both Linux servers and use SCP to copy the files between them:

s2/backups> **scp my.mbi\_2206\_1429 s3:/backups**

Restore the backup. Connect to the target host (the host for **s3** in this case), and restore the backup using MySQL Enterprise Backup. Here are the steps:

Stop the corrupted server, if it is still running. For example, on Linux distributions that use systemd:

s3> systemctl stop mysqld

Preserve the configuration file mysqld-auto.cnf, if it is found in the corrupted server's data directory, by copying it to a safe location outside of the data directory. This is for preserving the server's [Section 5.1.9.3, “Persisted System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#persisted-system-variables), which are needed later.

Delete all contents in the data directory of **s3**. For example:

s3> rm -rf /var/lib/mysql/\*

If the system variables [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), [**innodb\_log\_group\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_group_home_dir), and [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) point to any directories other than the data directory, they should also be made empty; otherwise, the restore operation fails.

Restore the backup of **s2** onto the host of **s3**. With this approach, we are rebuilding **s3** as a new member, for which we do not need or do not want to use the old binary and relay logs in the backup; therefore, if these logs have been included in your backup, exclude them using the [--skip-binlog](https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/backup-capacity-options.html#option_meb_skip-binlog) and [--skip-relaylog](https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/backup-capacity-options.html#option_meb_skip-relaylog) options:

s3> **mysqlbackup --defaults-file=/etc/my.cnf \**

**--datadir=/var/lib/mysql \**

**--backup-image=/backups/my.mbi\_2206\_1429 \**

**--backup-dir=/tmp/restore\_`date +%d%m\_%H%M` \**

**--skip-binlog --skip-relaylog \**

**copy-back-and-apply-log**

**Note**

If you have healthy binary log and relay logs in the backup that you can transfer onto the target host with no issues, you are recommended to follow the easier procedure as described in [Restoring a Failed Member](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-restore-failed-member) above.

Restore the *mysqld-auto.cnf* file for s3 (only required if s3 used persistent system variables). The settings for the [Section 5.1.9.3, “Persisted System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#persisted-system-variables) that were used to configure the failed member must be provided to the restored server. These settings are to be found in the mysqld-auto.cnf file of the failed server, which you should have preserved in step 2 above. Restore the file to the data directory of the restored server. See [Restoring Persisted System Variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-meb-restore-persisted-variables) on what to do if you do not have a copy of the file.

**Note**

Do NOT restore the corrupted server's auto.cnf file to the data directory of the new member—when the rebuilt **s3** joins the group as a new member, it is going to be assigned a new server UUID.

Start the restored server. For example, on Linux distributions that use systemd:

systemctl start mysqld

**Note**

If the server you are restoring is a primary member, perform the steps described in [Restoring a Primary Member](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-meb-restore-primary) before starting the restored server.

Reconfigure the restored member to join Group Replication. Connect to the restored server with a [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client and reset the source and replica information with the following commands:

mysql> **RESET MASTER;**

mysql> **RESET SLAVE ALL;**

Or from MySQL 8.0.22:

mysql> **RESET REPLICA ALL;**

For the restored server to be able to recover automatically using Group Replication's built-in mechanism for [distributed recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery), configure the server's [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) variable. To do this, use the backup\_gtid\_executed.sql file included in the backup of **s2**, which is usually restored under the restored member's data directory. Disable binary logging, use the backup\_gtid\_executed.sql file to configure [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed), and then re-enable binary logging by issuing the following statements with your [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client:

mysql> **SET SQL\_LOG\_BIN=OFF;**

mysql> **SOURCE *datadir*/backup\_gtid\_executed.sql**

mysql> **SET SQL\_LOG\_BIN=ON;**

Then, configure the [Group Replication user credentials](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-user-credentials) on the member:

mysql> **CHANGE MASTER TO MASTER\_USER='*rpl\_user*', MASTER\_PASSWORD='*password*' /**

**FOR CHANNEL 'group\_replication\_recovery';**

Or from MySQL 8.0.23:

mysql> **CHANGE REPLICATION SOURCE TO SOURCE\_USER='*rpl\_user*', SOURCE\_PASSWORD='*password*' /**

**FOR CHANNEL 'group\_replication\_recovery';**

Restart Group Replication. Issue the following command to the restored server with your [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client:

mysql> **START GROUP\_REPLICATION;**

Before the restored instance can become an online member of the group, it needs to apply any transactions that have happened to the group after the backup was taken; this is achieved using Group Replication's [distributed recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery) mechanism, and the process starts after the [START GROUP\_REPLICATION](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement has been issued. To check the member status of the restored instance, issue:

mysql> SELECT member\_host, member\_port, member\_state FROM performance\_schema.replication\_group\_members;

+-------------+-------------+--------------+

| member\_host | member\_port | member\_state |

+-------------+-------------+--------------+

| s3 | 3306 | RECOVERING |

| s2 | 3306 | ONLINE |

| s1 | 3306 | ONLINE |

+-------------+-------------+--------------+

This shows that **s3** is applying transactions to catch up with the group. Once it has caught up with the rest of the group, its **member\_state** changes to **ONLINE**:

mysql> SELECT member\_host, member\_port, member\_state FROM performance\_schema.replication\_group\_members;

+-------------+-------------+--------------+

| member\_host | member\_port | member\_state |

+-------------+-------------+--------------+

| s3 | 3306 | ONLINE |

| s2 | 3306 | ONLINE |

| s1 | 3306 | ONLINE |

+-------------+-------------+--------------+

**Note**

If the server you are restoring is a primary member, once it has gained synchrony with the group and become **ONLINE**, perform the steps described at the end of [Restoring a Primary Member](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-meb-restore-primary) to revert the configuration changes you had made to the server before you started it.

The member has now been restored to the group as a new member.

**Restoring Persisted System Variables.** **mysqlbackup** does not provide support for backing up or preserving [Section 5.1.9.3, “Persisted System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#persisted-system-variables)—the file mysqld-auto.cnf is not included in a backup. To start the restored member with its persisted variable settings, you need to do one of the following:

Preserve a copy of the mysqld-auto.cnf file from the corrupted server, and copy it to the restored server's data directory.

Copy the mysqld-auto.cnf file from another member of the group into the restored server's data directory, if that member has the same persisted system variable settings as the corrupted member.

After the restored server is started and before you restart Group Replication, set all the system variables manually to their persisted values through a [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client.

**Restoring a Primary Member.** If the restored member is a primary in the group, care must be taken to prevent writes to the restored database during the Group Replication distributed recovery process. Depending on how the group is accessed by clients, there is a possibility of DML statements being executed on the restored member once it becomes accessible on the network, prior to the member finishing its catch-up on the activities it has missed while off the group. To avoid this, before starting the restored server, configure the following system variables in the server option file:

group\_replication\_start\_on\_boot=OFF

super\_read\_only=ON

event\_scheduler=OFF

These settings ensure that the member becomes read-only at startup and that the event scheduler is turned off while the member is catching up with the group during the distributed recovery process. Adequate error handling must also be configured on the clients, as they are prevented temporarily from performing DML operations during this period on the restored member. Once the restore process is fully completed and the restored member is in-sync with the rest of the group, revert those changes; restart the event scheduler:

mysql> **SET global event\_scheduler=ON;**

Edit the following system variables in the member's option file, so things are correctly configured for the next startup:

group\_replication\_start\_on\_boot=ON

super\_read\_only=OFF

event\_scheduler=ON

## 18.6 Group Replication Security

[18.6.1 Group Replication IP Address Permissions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-ip-address-permissions)

[18.6.2 Securing Group Communication Connections with Secure Socket Layer (SSL)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl)

[18.6.3 Securing Distributed Recovery Connections](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-securing)

This section explains how to secure a group, securing the connections between members of a group, or by establishing a security perimeter using an IP address allowlist.

### 18.6.1 Group Replication IP Address Permissions

The Group Replication plugin lets you specify an allowlist of hosts from which an incoming Group Communication System connection can be accepted. If you specify an allowlist on a server s1, then when server s2 is establishing a connection to s1 for the purpose of engaging group communication, s1 first checks the allowlist before accepting the connection from s2. If s2 is in the allowlist, then s1 accepts the connection, otherwise s1 rejects the connection attempt by s2. From MySQL 8.0.22, the system variable [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) is used to specify the allowlist, and for releases before MySQL 8.0.22, the system variable [**group\_replication\_ip\_whitelist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_whitelist) is used. The new system variable works in the same way as the old system variable, only the terminology has changed.

If you do not specify an allowlist explicitly, the group communication engine (XCom) automatically scans active interfaces on the host, and identifies those with addresses on private subnetworks, together with the subnet mask that is configured for each interface. These addresses, and the **localhost** IP address for IPv4 and (from MySQL 8.0.14) IPv6 are used to create an automatic Group Replication allowlist. The automatic allowlist therefore includes any IP addresses that are found for the host in the following ranges after the appropriate subnet mask has been applied:

IPv4 (as defined in RFC 1918)

10/8 prefix (10.0.0.0 - 10.255.255.255) - Class A

172.16/12 prefix (172.16.0.0 - 172.31.255.255) - Class B

192.168/16 prefix (192.168.0.0 - 192.168.255.255) - Class C

IPv6 (as defined in RFC 4193 and RFC 5156)

fc00:/7 prefix - unique-local addresses

fe80::/10 prefix - link-local unicast addresses

127.0.0.1 - localhost for IPv4

::1 - localhost for IPv6

An entry is added to the error log stating the addresses that have been allowed automatically for the host.

The automatic allowlist of private addresses cannot be used for connections from servers outside the private network, so a server, even if it has interfaces on public IPs, does not by default allow Group Replication connections from external hosts. For Group Replication connections between server instances that are on different machines, you must provide public IP addresses and specify these as an explicit allowlist. If you specify any entries for the allowlist, the private and **localhost** addresses are not added automatically, so if you use any of these, you must specify them explicitly.

To specify an allowlist manually, use the [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) (from MySQL 8.0.22) or [**group\_replication\_ip\_whitelist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_whitelist) system variable. Before MySQL 8.0.24, you cannot change the allowlist on a server while it is an active member of a replication group. If the member is active, you must execute [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) before changing the allowlist, and [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) afterwards. From MySQL 8.0.24, you can change the allowlist while Group Replication is running.

The allowlist must contain the IP address or host name that is specified in each member's [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) system variable. This address is not the same as the MySQL server SQL protocol host and port, and is not specified in the [**bind\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_bind_address) system variable for the server instance. If a host name used as the Group Replication local address for a server instance resolves to both an IPv4 and an IPv6 address, the IPv4 address is preferred for Group Replication connections.

IP addresses specified as distributed recovery endpoints, and the IP address for the member's standard SQL client connection if that is used for distributed recovery (which is the default), do not need to be added to the allowlist. The allowlist is only for the address specified by [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) for each member. A joining member must have its initial connection to the group permitted by the allowlist in order to retrieve the address or addresses for distributed recovery.

In the allowlist, you can specify any combination of the following:

IPv4 addresses (for example, **198.51.100.44**)

IPv4 addresses with CIDR notation (for example, **192.0.2.21/24**)

IPv6 addresses, from MySQL 8.0.14 (for example, **2001:db8:85a3:8d3:1319:8a2e:370:7348**)

IPv6 addresses with CIDR notation, from MySQL 8.0.14 (for example, **2001:db8:85a3:8d3::/64**)

Host names (for example, **example.org**)

Host names with CIDR notation (for example, **www.example.com/24**)

Before MySQL 8.0.14, host names could only resolve to IPv4 addresses. From MySQL 8.0.14, host names can resolve to IPv4 addresses, IPv6 addresses, or both. If a host name resolves to both an IPv4 and an IPv6 address, the IPv4 address is always used for Group Replication connections. You can use CIDR notation in combination with host names or IP addresses to permit a block of IP addresses with a particular network prefix, but do ensure that all the IP addresses in the specified subnet are under your control.

**Note**

When a connection attempt from an IP address is refused because the address is not in the allowlist, the refusal message always prints the IP address in IPv6 format. IPv4 addresses are preceded by **::ffff:** in this format (an IPV4-mapped IPv6 address). You do not need to use this format to specify IPv4 addresses in the allowlist; use the standard IPv4 format for them.

You must stop and restart Group Replication on a member in order to change its allowlist. A comma must separate each entry in the allowlist. For example:

mysql> STOP GROUP\_REPLICATION;

mysql> SET GLOBAL group\_replication\_ip\_allowlist="192.0.2.21/24,198.51.100.44,203.0.113.0/24,2001:db8:85a3:8d3:1319:8a2e:370:7348,example.org,www.example.com/24";

mysql> START GROUP\_REPLICATION;

To join a replication group, a server needs to be permitted on the seed member to which it makes the request to join the group. Typically, this would be the bootstrap member for the replication group, but it can be any of the servers listed by the [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) option in the configuration for the server joining the group. If any of the seed members for the group are listed in the [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) option with an IPv6 address when a joining member has an IPv4 [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address), or the reverse, you must also set up and permit an alternative address for the joining member for the protocol offered by the seed member (or a host name that resolves to an address for that protocol). This is because when a server joins a replication group, it must make the initial contact with the seed member using the protocol that the seed member advertises in the [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) option, whether that is IPv4 or IPv6. If a joining member does not have a permitted address for the appropriate protocol, its connection attempt is refused. For more information on managing mixed IPv4 and IPv6 replication groups, see [Section 18.5.5, “Support For IPv6 And For Mixed IPv6 And IPv4 Groups”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-ipv6).

When a replication group is reconfigured (for example, when a new primary is elected or a member joins or leaves), the group members re-establish connections between themselves. If a group member is only permitted by servers that are no longer part of the replication group after the reconfiguration, it is unable to reconnect to the remaining servers in the replication group that do not permit it. To avoid this scenario entirely, specify the same allowlist for all servers that are members of the replication group.

**Note**

It is possible to configure different allowlists on different group members according to your security requirements, for example, in order to keep different subnets separate. If you need to configure different allowlists to meet your security requirements, ensure that there is sufficient overlap between the allowlists in the replication group to maximize the possibility of servers being able to reconnect in the absence of their original seed member.

For host names, name resolution takes place only when a connection request is made by another server. A host name that cannot be resolved is not considered for allowlist validation, and a warning message is written to the error log. Forward-confirmed reverse DNS (FCrDNS) verification is carried out for resolved host names.

**Warning**

Host names are inherently less secure than IP addresses in an allowlist. FCrDNS verification provides a good level of protection, but can be compromised by certain types of attack. Specify host names in your allowlist only when strictly necessary, and ensure that all components used for name resolution, such as DNS servers, are maintained under your control. You can also implement name resolution locally using the hosts file, to avoid the use of external components.

### 18.6.2 Securing Group Communication Connections with Secure Socket Layer (SSL)

Secure sockets can be used for group communication connections between members of a group. The Group Replication system variable [**group\_replication\_ssl\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ssl_mode) is used to activate the use of SSL for group communication connections and specify the security mode for the connections. The default setting means that SSL is not used. The option has the following possible values:

**Table 18.2 group\_replication\_ssl\_mode configuration values**

| **Value** | **Description** |
| --- | --- |
| DISABLED | Establish an unencrypted connection (the default). |
| REQUIRED | Establish a secure connection if the server supports secure connections. |
| VERIFY\_CA | Like REQUIRED, but additionally verify the server TLS certificate against the configured Certificate Authority (CA) certificates. |
| VERIFY\_IDENTITY | Like VERIFY\_CA, but additionally verify that the server certificate matches the host to which the connection is attempted. |

The remainder of the configuration for Group Replication's group communication connections is taken from the server's SSL configuration. For more information on the options for configuring the server SSL, see [Command Options for Encrypted Connections](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#encrypted-connection-options). The server SSL options that are applied to Group Replication's group communication connections are as follows:

**Table 18.3 SSL Options**

| **Server Configuration** | **Description** |
| --- | --- |
| [**ssl\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ssl_key) | The path name of the SSL private key file in PEM format. On the client side, this is the client private key. On the server side, this is the server private key. |
| [**ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ssl_cert) | The path name of the SSL public key certificate file in PEM format. On the client side, this is the client public key certificate. On the server side, this is the server public key certificate. |
| [**ssl\_ca**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ssl_ca) | The path name of the Certificate Authority (CA) certificate file in PEM format. |
| [**ssl\_capath**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ssl_capath) | The path name of the directory that contains trusted SSL certificate authority (CA) certificate files in PEM format. |
| [**ssl\_crl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ssl_crl) | The path name of the file containing certificate revocation lists in PEM format. |
| [**ssl\_crlpath**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ssl_crlpath) | The path name of the directory that contains certificate revocation list files in PEM format. |
| [**ssl\_cipher**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ssl_cipher) | A list of permissible ciphers for encrypted connections. |
| [**tls\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tls_version) | A list of the TLS protocols the server permits for encrypted connections. |
| [**tls\_ciphersuites**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tls_ciphersuites) | Which TLSv1.3 ciphersuites the server permits for encrypted connections. |

**Important**

Support for the TLSv1.3 protocol is available in MySQL Server as of MySQL 8.0.16, provided that MySQL was compiled using OpenSSL 1.1.1 or higher. Group Replication supports TLSv1.3 from MySQL 8.0.18. In MySQL 8.0.16 and MySQL 8.0.17, if the server supports TLSv1.3, the protocol is not supported in the group communication engine and cannot be used by Group Replication.

In the list of TLS protocols specified in the [**tls\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tls_version) system variable, ensure the specified versions are contiguous (for example, **TLSv1,TLSv1.1,TLSv1.2**). If there are any gaps in the list of protocols (for example, if you specified **TLSv1,TLSv1.2**, omitting TLS 1.1) Group Replication might be unable to make group communication connections.

In MySQL 8.0.18, TLSv1.3 can be used in Group Replication for the distributed recovery connection, but the [**group\_replication\_recovery\_tls\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_tls_version) and [**group\_replication\_recovery\_tls\_ciphersuites**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_tls_ciphersuites) system variables are not available. The donor servers must therefore permit the use of at least one TLSv1.3 ciphersuite that is enabled by default, as listed in [Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#encrypted-connection-protocols-ciphers). From MySQL 8.0.19, you can use the options to configure client support for any selection of ciphersuites, including only non-default ciphersuites if you want.

In a replication group, OpenSSL negotiates the use of the highest TLS protocol that is supported by all members. A joining member that is configured to use only TLSv1.3 ([**tls\_version=TLSv1.3**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tls_version)) cannot join a replication group where any existing member does not support TLSv1.3, because the group members in that case are using a lower TLS protocol version. To join the member to the group, you must configure the joining member to also permit the use of lower TLS protocol versions supported by the existing group members. Conversely, if a joining member does not support TLSv1.3, but the existing group members all do and are using that version for connections to each other, the member can join if the existing group members already permit the use of a suitable lower TLS protocol version, or if you configure them to do so. In that situation, OpenSSL uses a lower TLS protocol version for the connections from each member to the joining member. Each member's connections to other existing members continue to use the highest available protocol that both members support.

From MySQL 8.0.16, you can change the [**tls\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tls_version) system variable at runtime to alter the list of permitted TLS protocol versions for the server. Note that for Group Replication, the [**ALTER INSTANCE RELOAD TLS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-reload-tls) statement, which reconfigures the server's TLS context from the current values of the system variables that define the context, does not change the TLS context for Group Replication's group communication connection while Group Replication is running. To apply the reconfiguration to these connections, you must execute [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) followed by [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) to restart Group Replication on the member or members where you changed the [**tls\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tls_version) system variable. Similarly, if you want to make all members of a group change to using a higher or lower TLS protocol version, you must carry out a rolling restart of Group Replication on the members after changing the list of permitted TLS protocol versions, so that OpenSSL negotiates the use of the higher TLS protocol version when the rolling restart is completed. For instructions to change the list of permitted TLS protocol versions at runtime, see [Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#encrypted-connection-protocols-ciphers) and [Server-Side Runtime Configuration and Monitoring for Encrypted Connections](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#using-encrypted-connections-server-side-runtime-configuration).

The following example shows a section from a **my.cnf** file that configures SSL on a server, and activates SSL for Group Replication group communication connections:

[mysqld]

ssl\_ca = "cacert.pem"

ssl\_capath = "/.../ca\_directory"

ssl\_cert = "server-cert.pem"

ssl\_cipher = "DHE-RSA-AEs256-SHA"

ssl\_crl = "crl-server-revoked.crl"

ssl\_crlpath = "/.../crl\_directory"

ssl\_key = "server-key.pem"

group\_replication\_ssl\_mode= REQUIRED

**Important**

The [**ALTER INSTANCE RELOAD TLS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-reload-tls) statement, which reconfigures the server's TLS context from the current values of the system variables that define the context, does not change the TLS context for Group Replication's group communication connections while Group Replication is running. To apply the reconfiguration to these connections, you must execute [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) followed by [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) to restart Group Replication.

Connections made between a joining member and an existing member for distributed recovery are not covered by the options described above. These connections use Group Replication's dedicated distributed recovery SSL options, which are described in [Section 18.6.3.2, “Secure Socket Layer (SSL) Connections for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-configuring-ssl-for-recovery).

### 18.6.3 Securing Distributed Recovery Connections

[18.6.3.1 Secure User Credentials for Distributed Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-user)

[18.6.3.2 Secure Socket Layer (SSL) Connections for Distributed Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-configuring-ssl-for-recovery)

When a member joins the group, distributed recovery is carried out using a combination of a remote cloning operation, if available and appropriate, and an asynchronous replication connection. For a full description of distributed recovery, see [Section 18.5.3, “Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery).

The connection that an existing member offers to a joining member for distributed recovery is not the same connection that is used by Group Replication for communication between online members of the group. Up to MySQL 8.0.20, group members offer their standard SQL client connection to joining members for distributed recovery, as specified by MySQL Server's [**hostname**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_hostname) and [**port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_port) system variables. From MySQL 8.0.21, group members may advertise an alternative list of distributed recovery endpoints as dedicated client connections for joining members. For more details, see [Section 18.5.3.1, “Connections for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-connections).

To secure distributed recovery connections in the group, ensure that user credentials for the replication user are properly secured, and use SSL for distributed recovery connections if possible.

#### 18.6.3.1 Secure User Credentials for Distributed Recovery

State transfer from the binary log requires a replication user with the correct permissions so that Group Replication can establish direct member-to-member replication channels. The same replication user is used for distributed recovery on all the group members. If group members have been set up to support the use of a remote cloning operation as part of distributed recovery, which is available from MySQL 8.0.17, this replication user is also used as the clone user on the donor, and requires the correct permissions for this role too. For detailed instructions to set up this user, see [Section 18.2.1.3, “User Credentials For Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-user-credentials).

To secure the user credentials, you can require SSL for connections with the user account, and (from MySQL 8.0.21) you can provide the user credentials when Group Replication is started, rather than storing them in the replica status tables. Also, if you are using caching SHA-2 authentication, you must set up RSA key-pairs on the group members.

##### 18.6.3.1.1 Replication User With The Caching SHA-2 Authentication Plugin

By default, users created in MySQL 8 use [Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#caching-sha2-pluggable-authentication). If the replication user you configure for distributed recovery uses the caching SHA-2 authentication plugin, and you are not using SSL for distributed recovery connections, RSA key-pairs are used for password exchange. For more information on RSA key-pairs, see [Section 6.3.3, “Creating SSL and RSA Certificates and Keys”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#creating-ssl-rsa-files).

In this situation, you can either copy the public key of the **rpl\_user** to the joining member, or configure the donors to provide the public key when requested. The more secure approach is to copy the public key of the replication user account to the joining member. Then you need to configure the [**group\_replication\_recovery\_public\_key\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_public_key_path) system variable on the joining member with the path to the public key for the replication user account.

The less secure approach is to set [**group\_replication\_recovery\_get\_public\_key=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_get_public_key) on donors so that they provide the public key of the replication user account to joining members. There is no way to verify the identity of a server, therefore only set [**group\_replication\_recovery\_get\_public\_key=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_get_public_key) when you are sure there is no risk of server identity being compromised, for example by a man-in-the-middle attack.

##### 18.6.3.1.2 Replication User With SSL

A replication user that requires an SSL connection must be created before the server joining the group (the joining member) connects to the donor. Typically, this is set up at the time you are provisioning a server to join the group. To create a replication user for distributed recovery that requires an SSL connection, issue these statements on all servers that are going to participate in the group:

mysql> **SET SQL\_LOG\_BIN=0;**

mysql> **CREATE USER '*rec\_ssl\_user*'@'%' IDENTIFIED BY '*password*' REQUIRE SSL;**

mysql> **GRANT replication slave ON \*.\* TO '*rec\_ssl\_user*'@'%';**

mysql> **GRANT BACKUP\_ADMIN ON \*.\* TO '*rec\_ssl\_user*'@'%';**

mysql> **FLUSH PRIVILEGES;**

mysql> **SET SQL\_LOG\_BIN=1;**

##### 18.6.3.1.3 Providing Replication User Credentials Securely

To supply the user credentials for the replication user, you can set them permanently as the credentials for the **group\_replication\_recovery** channel, using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement. Alternatively, from MySQL 8.0.21, you can specify them on the [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement each time Group Replication is started. User credentials specified on [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) take precedence over any user credentials that have been set using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement.

User credentials set using [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) are stored in plain text in the replication metadata repositories on the server, but user credentials specified on [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) are saved in memory only, and are removed by a [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) statement or server shutdown. Using [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) to specify the user credentials therefore helps to secure the Group Replication servers against unauthorized access. However, this method is not compatible with starting Group Replication automatically, as specified by the [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) system variable.

If you want to set the user credentials permanently using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, issue this statement on the member that is going to join the group:

mysql> **CHANGE MASTER TO MASTER\_USER='*rec\_ssl\_user*', MASTER\_PASSWORD='*password*'**

**FOR CHANNEL 'group\_replication\_recovery';**

Or from MySQL 8.0.23:

mysql> **CHANGE REPLICATION SOURCE TO SOURCE\_USER='*rec\_ssl\_user*', SOURCE\_PASSWORD='*password*'**

**FOR CHANNEL 'group\_replication\_recovery';**

To supply the user credentials on [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication), issue this statement when starting Group Replication for the first time, or after a server restart:

mysql> START GROUP\_REPLICATION USER='***rec\_ssl\_user***', PASSWORD='***password***';

**Important**

If you switch to using [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) to specify user credentials on a server that previously supplied the credentials using [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to), you must complete the following steps to get the security benefits of this change.

Stop Group Replication on the group member using a [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) statement. Although it is possible to take the following two steps while Group Replication is running, you need to restart Group Replication to implement the changes.

Set the value of the [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) system variable to **OFF** (the default is **ON**).

Remove the distributed recovery credentials from the replica status tables by issuing this statement:

mysql> **CHANGE MASTER TO MASTER\_USER='', MASTER\_PASSWORD=''**

**FOR CHANNEL 'group\_replication\_recovery';**

Or from MySQL 8.0.23:

mysql> **CHANGE REPLICATION SOURCE TO SOURCE\_USER='', SOURCE\_PASSWORD=''**

**FOR CHANNEL 'group\_replication\_recovery';**

Restart Group Replication on the group member using a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement that specifies the distributed recovery user credentials.

Without these steps, the credentials remain stored in the replica status tables, and can also be transferred to other group members during remote cloning operations for distributed recovery. The **group\_replication\_recovery** channel could then be inadvertently started with the stored credentials, on either the original member or members that were cloned from it. An automatic start of Group Replication on server boot (including after a remote cloning operation) would use the stored user credentials, and they would also be used if an operator did not specify the distributed recovery credentials on a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) command.

#### 18.6.3.2 Secure Socket Layer (SSL) Connections for Distributed Recovery

Whether the distributed recovery connection is made using the standard SQL client connection or a distributed recovery endpoint, to configure the connection securely, you can use Group Replication's dedicated distributed recovery SSL options. These options correspond to the server SSL options that are used for group communication connections, but they are only applied for distributed recovery connections. By default, distributed recovery connections do not use SSL, even if you activated SSL for group communication connections, and the server SSL options are not applied for distributed recovery connections. You must configure these connections separately.

If a remote cloning operation is used as part of distributed recovery, Group Replication automatically configures the clone plugin's SSL options to match your settings for the distributed recovery SSL options. (For details of how the clone plugin uses SSL, see [Configuring an Encrypted Connection for Cloning](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin-remote-ssl).)

The distributed recovery SSL options are as follows:

[**group\_replication\_recovery\_use\_ssl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_use_ssl): Set to **ON** to make Group Replication use SSL for distributed recovery connections, including remote cloning operations and state transfer from a donor's binary log. You can just set this option and none of the other distributed recovery SSL options, in which case the server automatically generates certificates to use for the connection, and uses the default cipher suites. If you want to configure the certificates and cipher suites for the connection, use the other distributed recovery SSL options to do this.

[**group\_replication\_recovery\_ssl\_ca**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_ca): The path name of the Certificate Authority (CA) file to use for distributed recovery connections. Group Replication automatically configures the clone SSL option [**clone\_ssl\_ca**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_ssl_ca) to match this.

[**group\_replication\_recovery\_ssl\_capath**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_capath): The path name of a directory that contains trusted SSL certificate authority (CA) certificate files.

[**group\_replication\_recovery\_ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_cert): The path name of the SSL public key certificate file to use for distributed recovery connections. Group Replication automatically configures the clone SSL option [**clone\_ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_ssl_cert) to match this.

[**group\_replication\_recovery\_ssl\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_key): The path name of the SSL private key file to use for distributed recovery connections. Group Replication automatically configures the clone SSL option [**clone\_ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_ssl_cert) to match this.

[**group\_replication\_recovery\_ssl\_verify\_server\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_verify_server_cert): Makes the distributed recovery connection check the server's Common Name value in the donor sent certificate. Setting this option to **ON** is the equivalent for distributed recovery connections of setting **VERIFY\_IDENTITY** for the [**group\_replication\_ssl\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ssl_mode) option for group communication connections.

[**group\_replication\_recovery\_ssl\_crl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_crl): The path name of a file containing certificate revocation lists.

[**group\_replication\_recovery\_ssl\_crlpath**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_crlpath): The path name of a directory containing certificate revocation lists.

[**group\_replication\_recovery\_ssl\_cipher**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_cipher): A list of permissible ciphers for connection encryption for the distributed recovery connection. Specify a list of one or more cipher names, separated by colons. For information about which encryption ciphers MySQL supports, see [Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#encrypted-connection-protocols-ciphers).

[**group\_replication\_recovery\_tls\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_tls_version): A comma-separated list of one or more permitted TLS protocols for connection encryption when this server instance is the client in the distributed recovery connection, that is, the joining member. Ensure the specified versions are contiguous (for example, “**TLSv1,TLSv1.1,TLSv1.2**”). If this system variable is not set, the default “**TLSv1,TLSv1.1,TLSv1.2,TLSv1.3**” is used. The group members involved in each distributed recovery connection as the client (joining member) and server (donor) negotiate the highest protocol version that they are both set up to support. This system variable is available from MySQL 8.0.19.

[**group\_replication\_recovery\_tls\_ciphersuites**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_tls_ciphersuites): A colon-separated list of one or more permitted ciphersuites when TLSv1.3 is used for connection encryption for the distributed recovery connection, and this server instance is the client in the distributed recovery connection, that is, the joining member. If this system variable is set to **NULL** when TLSv1.3 is used (which is the default if you do not set the system variable), the ciphersuites that are enabled by default are allowed, as listed in [Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#encrypted-connection-protocols-ciphers). If this system variable is set to the empty string, no cipher suites are allowed, and TLSv1.3 are therefore not used. This system variable is available beginning with MySQL 8.0.19.

## 18.7 Group Replication Performance

[18.7.1 Fine Tuning the Group Communication Thread](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-fine-tuning-the-group-communication-thread)

[18.7.2 Flow Control](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-flow-control)

[18.7.3 Message Compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-message-compression)

[18.7.4 Message Fragmentation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-performance-message-fragmentation)

[18.7.5 XCom Cache Management](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-performance-xcom-cache)

[18.7.6 Responses to Failure Detection and Network Partitioning](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-responses-failure)

This section explains how to use the available configuration options to gain the best performance from your replication group.

### 18.7.1 Fine Tuning the Group Communication Thread

The group communication thread (GCT) runs in a loop while the Group Replication plugin is loaded. The GCT receives messages from the group and from the plugin, handles quorum and failure detection related tasks, sends out some keep alive messages and also handles the incoming and outgoing transactions from/to the server/group. The GCT waits for incoming messages in a queue. When there are no messages, the GCT waits. By configuring this wait to be a little longer (doing an active wait) before actually going to sleep can prove to be beneficial in some cases. This is because the alternative is for the operating system to switch out the GCT from the processor and do a context switch.

To force the GCT to do an active wait, use the [**group\_replication\_poll\_spin\_loops**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_poll_spin_loops) option, which makes the GCT loop, doing nothing relevant for the configured number of loops, before actually polling the queue for the next message.

For example:

mysql> SET GLOBAL group\_replication\_poll\_spin\_loops= 10000;

### 18.7.2 Flow Control

[18.7.2.1 Probes and Statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-probes-and-statistics)

[18.7.2.2 Group Replication Throttling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-throttling)

Group Replication ensures that a transaction only commits after a majority of the members in a group have received it and agreed on the relative order between all transactions that were sent concurrently. This approach works well if the total number of writes to the group does not exceed the write capacity of any member in the group. If it does and some of the members have less write throughput than others, particularly less than the writer members, those members can start lagging behind of the writers.

Having some members lagging behind the group brings some problematic consequences, particularly, the reads on such members may externalize very old data. Depending on why the member is lagging behind, other members in the group may have to save more or less replication context to be able to fulfil potential data transfer requests from the slow member.

There is however a mechanism in the replication protocol to avoid having too much distance, in terms of transactions applied, between fast and slow members. This is known as the flow control mechanism. It tries to address several goals:

to keep the members close enough to make buffering and de-synchronization between members a small problem;

to adapt quickly to changing conditions like different workloads or more writers in the group;

to give each member a fair share of the available write capacity;

to not reduce throughput more than strictly necessary to avoid wasting resources.

Given the design of Group Replication, the decision whether to throttle or not may be decided taking into account two work queues: (i) the certification queue; (ii) and on the binary log applier queue. Whenever the size of one of these queues exceeds the user-defined threshold, the throttling mechanism is triggered. Only configure: (i) whether to do flow control at the certifier or at the applier level, or both; and (ii) what is the threshold for each queue.

The flow control depends on two basic mechanisms:

the monitoring of members to collect some statistics on throughput and queue sizes of all group members to make educated guesses on what is the maximum write pressure each member should be subjected to;

the throttling of members that are trying to write beyond their fair-share of the available capacity at each moment in time.

#### 18.7.2.1 Probes and Statistics

The monitoring mechanism works by having each member deploying a set of probes to collect information about its work queues and throughput. It then propagates that information to the group periodically to share that data with the other members.

Such probes are scattered throughout the plugin stack and allow one to establish metrics, such as:

the certifier queue size;

the replication applier queue size;

the total number of transactions certified;

the total number of remote transactions applied in the member;

the total number of local transactions.

Once a member receives a message with statistics from another member, it calculates additional metrics regarding how many transactions were certified, applied and locally executed in the last monitoring period.

Monitoring data is shared with others in the group periodically. The monitoring period must be high enough to allow the other members to decide on the current write requests, but low enough that it has minimal impact on group bandwidth. The information is shared every second, and this period is sufficient to address both concerns.

#### 18.7.2.2 Group Replication Throttling

Based on the metrics gathered across all servers in the group, a throttling mechanism kicks in and decides whether to limit the rate a member is able to execute/commit new transactions.

Therefore, metrics acquired from all members are the basis for calculating the capacity of each member: if a member has a large queue (for certification or the applier thread), then the capacity to execute new transactions should be close to ones certified or applied in the last period.

The lowest capacity of all the members in the group determines the real capacity of the group, while the number of local transactions determines how many members are writing to it, and, consequently, how many members should that available capacity be shared with.

This means that every member has an established write quota based on the available capacity, in other words a number of transactions it can safely issue for the next period. The writer-quota is enforced by the throttling mechanism if the queue size of the certifier or the binary log applier exceeds a user-defined threshold.

The quota is reduced by the number of transactions that were delayed in the last period, and then also further reduced by 10% to allow the queue that triggered the problem to reduce its size. In order to avoid large jumps in throughput once the queue size goes beyond the threshold, the throughput is only allowed to grow by the same 10% per period after that.

The current throttling mechanism does not penalize transactions below quota, but delays finishing those transactions that exceed it until the end of the monitoring period. As a consequence, if the quota is very small for the write requests issued some transactions may have latencies close to the monitoring period.

### 18.7.3 Message Compression

For messages sent between online group members, Group Replication enables message compression by default. Whether a specific message is compressed depends on the threshold that you configure using the [**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold) system variable. Messages that have a payload larger than the specified number of bytes are compressed.

The default compression threshold is 1000000 bytes. You could use the following statements to increase the compression threshold to 2MB, for example:

STOP GROUP\_REPLICATION;

SET GLOBAL group\_replication\_compression\_threshold = 2097152;

START GROUP\_REPLICATION;

If you set [**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold) to zero, message compression is disabled.

Group Replication uses the LZ4 compression algorithm to compress messages sent in the group. Note that the maximum supported input size for the LZ4 compression algorithm is 2113929216 bytes. This limit is lower than the maximum possible value for the [**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold) system variable, which is matched to the maximum message size accepted by XCom. The LZ4 maximum input size is therefore a practical limit for message compression, and transactions above this size cannot be committed when message compression is enabled. With the LZ4 compression algorithm, do not set a value greater than 2113929216 bytes for [**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold).

The value of [**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold) is not required by Group Replication to be the same on all group members. However, it is advisable to set the same value on all group members in order to avoid unnecessary rollback of transactions, failure of message delivery, or failure of message recovery.

From MySQL 8.0.18, you can also configure compression for messages sent for distributed recovery by the method of state transfer from a donor's binary log. Compression for these messages, which are sent from a donor already in the group to a joining member, is controlled separately using the [**group\_replication\_recovery\_compression\_algorithms**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_compression_algorithms) and [**group\_replication\_recovery\_zstd\_compression\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_zstd_compression_level) system variables. For more information, see [Section 4.2.8, “Connection Compression Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#connection-compression-control).

Binary log transaction compression (available as of MySQL 8.0.20), which is activated by the [**binlog\_transaction\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression) system variable, can also be used to save bandwidth. The transaction payloads remain compressed when they are transferred between group members. If you use binary log transaction compression in combination with Group Replication's message compression, message compression has less opportunity to act on the data, but can still compress headers and those events and transaction payloads that are uncompressed. For more information on binary log transaction compression, see [Section 5.4.4.5, “Binary Log Transaction Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-transaction-compression).

Compression for messages sent in the group happens at the group communication engine level, before the data is handed over to the group communication thread, so it takes place within the context of the **mysql** user session thread. If the message payload size exceeds the threshold set by [**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold), the transaction payload is compressed before being sent out to the group, and decompressed when it is received. Upon receiving a message, the member checks the message envelope to verify whether it is compressed or not. If needed, then the member decompresses the transaction, before delivering it to the upper layer. This process is shown in the following figure.

**Figure 18.15 Compression Support**

When network bandwidth is a bottleneck, message compression can provide up to 30-40% throughput improvement at the group communication level. This is especially important within the context of large groups of servers under load. The TCP peer-to-peer nature of the interconnections between N participants in the group makes the sender send the same amount of data N times. Furthermore, binary logs are likely to exhibit a high compression ratio. This makes compression a compelling feature for Group Replication workloads that contain large transactions.

### 18.7.4 Message Fragmentation

When an abnormally large message is sent between Group Replication group members, it can result in some group members being reported as failed and expelled from the group. This is because the single thread used by Group Replication's group communication engine (XCom, a Paxos variant) is occupied processing the message for too long, so some of the group members might report the receiver as failed. From MySQL 8.0.16, by default, large messages are automatically split into fragments that are sent separately and reassembled by the recipients.

The system variable [**group\_replication\_communication\_max\_message\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_communication_max_message_size) specifies a maximum message size for Group Replication communications, above which messages are fragmented. The default maximum message size is 10485760 bytes (10 MiB). The greatest permitted value is the same as the maximum value of the [**slave\_max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_max_allowed_packet) system variable, which is 1073741824 bytes (1 GB). The setting for [**group\_replication\_communication\_max\_message\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_communication_max_message_size) must be less than the [**slave\_max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_max_allowed_packet) setting, because the applier thread cannot handle message fragments larger than [**slave\_max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_max_allowed_packet). To switch off fragmentation, specify a zero value for [**group\_replication\_communication\_max\_message\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_communication_max_message_size).

As with most other Group Replication system variables, you must restart the Group Replication plugin for the change to take effect. For example:

STOP GROUP\_REPLICATION;

SET GLOBAL group\_replication\_communication\_max\_message\_size= 5242880;

START GROUP\_REPLICATION;

Message delivery for a fragmented message is considered complete when all the fragments of the message have been received and reassembled by all the group members. Fragmented messages include information in their headers that enables a member joining during message transmission to recover the earlier fragments that were sent before it joined. If the joining member fails to recover the fragments, it expels itself from the group.

In order for a replication group to use fragmentation, all group members must be at MySQL 8.0.16 or above, and the Group Replication communication protocol version in use by the group must allow fragmentation. You can inspect the communication protocol in use by a group by using the [**group\_replication\_get\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-get-communication-protocol) UDF, which returns the oldest MySQL Server version that the group supports. Versions from MySQL 5.7.14 allow compression of messages, and versions from MySQL 8.0.16 also allow fragmentation of messages. If all group members are at MySQL 8.0.16 or above and there is no requirement to allow members at earlier releases to join, you can use the [**group\_replication\_set\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-communication-protocol) UDF to set the communication protocol version to MySQL 8.0.16 or above in order to allow fragmentation. For more information, see [Section 18.5.1.4, “Setting a Group's Communication Protocol Version”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-communication-protocol).

If a replication group cannot use fragmentation because some members do not support it, the system variable [**group\_replication\_transaction\_size\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_transaction_size_limit) can be used to limit the maximum size of transactions the group accepts. In MySQL 8.0, the default setting is approximately 143 MB. Transactions above this size are rolled back. You can also use the system variable [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) to allow additional time (up to an hour) before a member under suspicion of having failed is expelled from the group.

### 18.7.5 XCom Cache Management

[18.7.5.1 Increasing the cache size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-performance-xcom-cache-increase)

[18.7.5.2 Reducing the cache size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-performance-xcom-cache-reduce)

The group communication engine for Group Replication (XCom, a Paxos variant) includes a cache for messages (and their metadata) exchanged between the group members as a part of the consensus protocol. Among other functions, the message cache is used for recovery of missed messages by members that reconnect with the group after a period where they were unable to communicate with the other group members.

From MySQL 8.0.16, a cache size limit can be set for XCom's message cache using the [**group\_replication\_message\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_message_cache_size) system variable. If the cache size limit is reached, XCom removes the oldest entries that have been decided and delivered. The same cache size limit should be set on all group members, because an unreachable member that is attempting to reconnect selects any other member at random for recovery of missed messages. The same messages should therefore be available in each member's cache.

Before MySQL 8.0.16, the cache size was 1 GB, and the default setting for the cache size from MySQL 8.0.16 is the same. Ensure that sufficient memory is available on your system for your chosen cache size limit, considering the size of MySQL Server's other caches and object pools. Note that the limit set using [**group\_replication\_message\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_message_cache_size) applies only to the data stored in the cache, and the cache structures require an additional 50 MB of memory.

When selecting a [**group\_replication\_message\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_message_cache_size) setting, do so with reference to the expected volume of messages in the time period before a member is expelled. The length of this time period is controlled by the [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) system variable, which determines the waiting period (up to an hour) that is allowed in addition to the initial 5-second detection period for members to return to the group rather than being expelled. Note that before MySQL 8.0.21, this time period defaulted to 5 seconds from the member becoming unavailable, which is just the detection period before a suspicion is created, because the additional expel timeout set by the [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) system variable defaulted to zero. From 8.0.21 the expel timeout defaults to 5 seconds, so by default a member is not expelled until it has been absent for at least 10 seconds.

#### 18.7.5.1 Increasing the cache size

If a member is absent for a period that is not long enough for it to be expelled from the group, it can reconnect and start participating in the group again by retrieving missed transactions from another member's XCom message cache. However, if the transactions that happened during the member's absence have been deleted from the other members' XCom message caches because their maximum size limit was reached, the member cannot reconnect in this way.

Group Replication's Group Communication System (GCS) alerts you, by a warning message, when a message that is likely to be needed for recovery by a member that is currently unreachable is removed from the message cache. This warning message is logged on all the active group members (only once for each unreachable member). Although the group members cannot know for sure what message was the last message seen by the unreachable member, the warning message indicates that the cache size might not be sufficient to support your chosen waiting period before a member is expelled.

In this situation, consider increasing the [**group\_replication\_message\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_message_cache_size) limit with reference to the expected volume of messages in the time period specified by the [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) system variable plus the 5-second detection period, so that the cache contains all the missed messages required for members to return successfully. You can also consider increasing the cache size limit temporarily if you expect a member to become unreachable for an unusual period of time.

#### 18.7.5.2 Reducing the cache size

The minimum setting for the XCom message cache size is 1 GB up to MySQL 8.0.20. From MySQL 8.0.21, the minimum setting is 134217728 bytes (128 MB), which enables deployment on a host that has a restricted amount of available memory. Having a very low [**group\_replication\_message\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_message_cache_size) setting is not recommended if the host is on an unstable network, because a smaller message cache makes it harder for group members to reconnect after a transient loss of connectivity.

If a reconnecting member cannot retrieve all the messages it needs from the XCom message cache, the member must leave the group and rejoin it, in order to retrieve the missing transactions from another member's binary log using distributed recovery. From MySQL 8.0.21, a member that has left a group makes three auto-rejoin attempts by default, so the process of rejoining the group can still take place without operator intervention. However, rejoining using distributed recovery is a significantly longer and more complex process than retrieving messages from an XCom message cache, so the member takes longer to become available and the performance of the group can be impacted. On a stable network, which minimizes the frequency and duration of transient losses of connectivity for members, the frequency of this occurrence should also be minimized, so the group might be able to tolerate a smaller XCom message cache size without a significant impact on its performance.

If you are considering reducing the cache size limit, you can query the Performance Schema table [**memory\_summary\_global\_by\_event\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-memory-summary-tables) using the following statement:

SELECT \* FROM performance\_schema.memory\_summary\_global\_by\_event\_name

WHERE EVENT\_NAME LIKE 'memory/group\_rpl/GCS\_XCom::xcom\_cache';

This returns memory usage statistics for the message cache, including the current number of cached entries and current size of the cache. If you reduce the cache size limit, XCom removes the oldest entries that have been decided and delivered until the current size is below the limit. XCom might temporarily exceed the cache size limit while this removal process is ongoing.

### 18.7.6 Responses to Failure Detection and Network Partitioning

[18.7.6.1 Expel Timeout](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-responses-failure-expel)

[18.7.6.2 Unreachable Majority Timeout](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-responses-failure-partition)

[18.7.6.3 Auto-Rejoin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-responses-failure-rejoin)

[18.7.6.4 Exit Action](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-responses-failure-exit)

Group Replication's failure detection mechanism is designed to identify group members that are no longer communicating with the group, and expel them as and when it seems likely that they have failed. Having a failure detection mechanism increases the chance that the group contains a majority of correctly working members, and that requests from clients are therefore processed correctly.

Normally, all group members regularly exchange messages with all other group members. If a group member does not receive any messages from a particular fellow member for 5 seconds, when this detection period ends, it creates a suspicion of the fellow member. When a suspicion times out, the suspected member is assumed to have failed, and is expelled from the group. An expelled member is removed from the membership list seen by the other members, but it does not know that it has been expelled from the group, so it sees itself as online and the other members as unreachable. If the member has not in fact failed (for example, because it was just disconnected due to a temporary network issue) and it is able to resume communication with the other members, it receives a view containing the information that it has been expelled from the group.

The responses of group members, including the failed member itself, to these situations can be configured at a number of points in the process. By default, the following behaviors happen if a member is suspected of having failed:

Up to MySQL 8.0.20, when a suspicion is created, it times out immediately. The suspected member is liable for expulsion as soon as the expired suspicion is identified by the group. The member could potentially survive for a further few seconds after the timeout because the check for expired suspicions is carried out periodically. From MySQL 8.0.21, a waiting period of 5 seconds is added before the suspicion times out and the suspected member is liable for expulsion.

If an expelled member resumes communication and realises that it was expelled, up to MySQL 8.0.20, it does not try to rejoin the group. From MySQL 8.0.21, it makes three automatic attempts to rejoin the group (with 5 minutes between each attempt), and if this auto-rejoin procedure does not work, it then stops trying to rejoin the group.

When an expelled member is not trying to rejoin the group, it switches to super read only mode and awaits operator attention. (The exception is in releases from MySQL 8.0.12 to 8.0.15, where the default was for the member to shut itself down. From MySQL 8.0.16, the behavior was changed to match the behavior in MySQL 5.7.)

You can use the Group Replication configuration options described in this section to change these behaviors either permanently or temporarily, to suit your system's requirements and your priorities. If you are experiencing unnecessary expulsions caused by slower networks or machines, networks with a high rate of unexpected transient outages, or planned network outages, consider increasing the expel timeout and auto-rejoin attempts. From MySQL 8.0.21, the default settings have been changed in this direction to reduce the frequency of the need for operator intervention to reinstate expelled members in these situations. Note that while a member is undergoing any of the default behaviors described above, although it does not accept writes, reads can still be made if the member is still communicating with clients, with an increasing likelihood of stale reads over time. If avoiding stale reads is a higher priority for you than avoiding operator intervention, consider reducing the expel timeout and auto-rejoin attempts or setting them to zero.

Members that have not failed might lose contact with part, but not all, of the replication group due to a network partition. For example, in a group of 5 servers (S1,S2,S3,S4,S5), if there is a disconnection between (S1,S2) and (S3,S4,S5) there is a network partition. The first group (S1,S2) is now in a minority because it cannot contact more than half of the group. Any transactions that are processed by the members in the minority group are blocked, because the majority of the group is unreachable, therefore the group cannot achieve quorum. For a detailed description of this scenario, see [Section 18.5.4, “Network Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-network-partitioning). In this situation, the default behavior is for the members in both the minority and the majority to remain in the group, continue to accept transactions (although they are blocked on the members in the minority), and wait for operator intervention. This behavior is also configurable.

Note that where group members are at an older MySQL Server release that does not support a relevant setting, or at a release with a different default, they act towards themselves and other group members according to the default behaviors stated above. For example, a member that does not support the [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) system variable expels other members as soon as an expired suspicion is detected, and this expulsion is accepted by other members even if they support the system variable and have a longer timeout set.

#### 18.7.6.1 Expel Timeout

You can use the [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) system variable, which is available from MySQL 8.0.13, to allow additional time between the creation of a suspicion and the expulsion of the suspect member. A suspicion is created when one server does not receive messages from another server, as explained in [Section 18.1.4.2, “Failure Detection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-failure-detection).

There is an initial 5-second detection period before a Group Replication group member creates a suspicion of another member (or of itself). A group member is then expelled when another member's suspicion of it (or its own suspicion of itself) times out. A further short period of time might elapse after that before the expelling mechanism detects and implements the expulsion. [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) specifies the period of time in seconds, called the expel timeout, that a group member waits between creating a suspicion, and expelling the suspected member. Suspect members are listed as **UNREACHABLE** during this waiting period, but are not removed from the group's membership list.

If a suspect member becomes active again before the suspicion times out at the end of the waiting period, the member applies all the messages that were buffered by the remaining group members in XCom's message cache and enters **ONLINE** state, without operator intervention. In this situation, the member is considered by the group as the same incarnation.

If a suspect member becomes active only after the suspicion times out and is able to resume communications, it receives a view where it is expelled and at that point realises it was expelled. You can use the [**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries) system variable, which is available from MySQL 8.0.16, to make the member automatically try to rejoin the group at this point. From MySQL 8.0.21, this feature is activated by default and the member makes three auto-rejoin attempts. If the auto-rejoin procedure does not succeed or is not attempted, the expelled member then follows the exit action specified by [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action).

The waiting period before expelling a member only applies to members that have previously been active in the group. Non-members that were never active in the group do not get this waiting period and are removed after the initial detection period because they took too long to join.

If [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) is set to 0, there is no waiting period, and a suspected member is liable for expulsion immediately after the 5-second detection period ends. This setting is the default up to and including MySQL 8.0.20. This is also the behavior of a group member which is at a MySQL Server version that does not support the [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) system variable. From MySQL 8.0.21, the value defaults to 5, meaning that a suspected member is liable for expulsion 5 seconds after the 5-second detection period. It is not mandatory for all members of a group to have the same setting for [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout), but it is recommended in order to avoid unexpected expulsions. Any member can create a suspicion of any other member, including itself, so the effective expel timeout is that of the member with the lowest setting.

Consider increasing the value of [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) from the default in the following scenarios:

The network is slow and the default 5 or 10 seconds before expulsion is not long enough for group members to always exchange at least one message.

The network sometimes has transient outages and you want to avoid unnecessary expulsions and primary member changes at these times.

The network is not under your direct control and you want to minimize the need for operator intervention.

A temporary network outage is expected and you do not want some or all of the members to be expelled due to this.

An individual machine is experiencing a slowdown and you do not want it to be expelled from the group.

You can specify an expel timeout up to a maximum of 3600 seconds (1 hour). It is important to ensure that XCom's message cache is sufficiently large to contain the expected volume of messages in your specified time period, plus the initial 5-second detection period, otherwise members cannot reconnect. You can adjust the cache size limit using the [**group\_replication\_message\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_message_cache_size) system variable. For more information, see [Section 18.7.5, “XCom Cache Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-performance-xcom-cache).

If any members in a group are currently under suspicion, the group membership cannot be reconfigured (by adding or removing members or electing a new leader). If group membership changes need to be implemented while one or more members are under suspicion, and you want the suspect members to remain in the group, take any actions required to make the members active again, if that is possible. If you cannot make the members active again and you want them to be expelled from the group, you can force the suspicions to time out immediately. Do this by changing the value of [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) on any active members to a value lower than the time that has already elapsed since the suspicions were created. The suspect members then become liable for expulsion immediately.

If a replication group member stops unexpectedly and is immediately restarted (for example, because it was started with **mysqld\_safe**), it automatically attempts to rejoin the group if [**group\_replication\_start\_on\_boot=on**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) is set. In this situation, it is possible for the restart and rejoin attempt to take place before the member's previous incarnation has been expelled from the group, in which case the member cannot rejoin. From MySQL 8.0.19, Group Replication automatically uses a Group Communication System (GCS) feature to retry the rejoin attempt for the member 10 times, with a 5-second interval between each retry. This should cover most cases and allow enough time for the previous incarnation to be expelled from the group, letting the member rejoin. Note that if the [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) system variable is set to specify a longer waiting period before the member is expelled, the automatic rejoin attempts might still not succeed.

For alternative mitigation strategies to avoid unnecessary expulsions where the [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) system variable is not available, see [Section 18.3.2, “Group Replication Limitations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-limitations).

#### 18.7.6.2 Unreachable Majority Timeout

By default, members that find themselves in a minority due to a network partition do not automatically leave the group. You can use the system variable [**group\_replication\_unreachable\_majority\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_unreachable_majority_timeout) to set a number of seconds for a member to wait after losing contact with the majority of group members, and then exit the group. Setting a timeout means you do not need to pro-actively monitor for servers that are in a minority group after a network partition, and you can avoid the possibility of creating a split-brain situation (with two versions of the group membership) due to inappropriate intervention.

When the timeout specified by [**group\_replication\_unreachable\_majority\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_unreachable_majority_timeout) elapses, all pending transactions that have been processed by the member and the others in the minority group are rolled back, and the servers in that group move to the **ERROR** state. You can use the [**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries) system variable, which is available from MySQL 8.0.16, to make the member automatically try to rejoin the group at this point. From MySQL 8.0.21, this feature is activated by default and the member makes three auto-rejoin attempts. If the auto-rejoin procedure does not succeed or is not attempted, the minority member then follows the exit action specified by [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action).

Consider the following points when deciding whether or not to set an unreachable majority timeout:

In a symmetric group, for example a group with two or four servers, if both partitions contain an equal number of servers, both groups consider themselves to be in a minority and enter the **ERROR** state. In this situation, the group has no functional partition.

While a minority group exists, any transactions processed by the minority group are accepted, but blocked because the minority servers cannot reach quorum, until either [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) is issued on those servers or the unreachable majority timeout is reached.

If you do not set an unreachable majority timeout, the servers in the minority group never enter the **ERROR** state automatically, and you must stop them manually.

Setting an unreachable majority timeout has no effect if it is set on the servers in the minority group after the loss of majority has been detected.

If you do not use the [**group\_replication\_unreachable\_majority\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_unreachable_majority_timeout)system variable, the process for operator invention in the event of a network partition is described in [Section 18.5.4, “Network Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-network-partitioning). The process involves checking which servers are functioning and forcing a new group membership if necessary.

#### 18.7.6.3 Auto-Rejoin

The [**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries) system variable, which is available from MySQL 8.0.16, makes a member that has been expelled or reached its unreachable majority timeout try to rejoin the group automatically. Up to MySQL 8.0.20, the value of the system variable defaults to 0, so auto-rejoin is not activated by default. From MySQL 8.0.21, the value of the system variable defaults to 3, meaning that the member automatically makes 3 attempts to rejoin the group, with 5 minutes between each.

When auto-rejoin is not activated, a member accepts its expulsion as soon as it resumes communication, and proceeds to the action specified by the [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action) system variable. After this, manual intervention is needed to bring the member back into the group. Using the auto-rejoin feature is appropriate if you can tolerate the possibility of stale reads and want to minimize the need for manual intervention, especially where transient network issues fairly often result in the expulsion of members.

With auto-rejoin, when the member's expulsion or unreachable majority timeout is reached, it makes an attempt to rejoin (using the current plugin option values), then continues to make further auto-rejoin attempts up to the specified number of tries. After an unsuccessful auto-rejoin attempt, the member waits 5 minutes before the next try. The auto-rejoin attempts and the time between them are called the auto-rejoin procedure. If the specified number of tries is exhausted without the member rejoining or being stopped, the member proceeds to the action specified by the [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action) system variable.

During and between auto-rejoin attempts, a member remains in super read only mode and displays an **ERROR** state on its view of the replication group. During this time, the member does not accept writes. However, reads can still be made on the member, with an increasing likelihood of stale reads over time. If you do want to intervene to take the member offline during the auto-rejoin procedure, the member can be stopped manually at any time by using a [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) statement or shutting down the server. If you cannot tolerate the possibility of stale reads for any period of time, set the [**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries) system variable to 0.

You can monitor the auto-rejoin procedure using the Performance Schema. While an auto-rejoin procedure is taking place, the Performance Schema table [**events\_stages\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-current-table) shows the event “Undergoing auto-rejoin procedure”, with the number of retries that have been attempted so far during this instance of the procedure (in the **WORK\_COMPLETED** field). The [**events\_stages\_summary\_global\_by\_event\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-stage-summary-tables) table shows the number of times the server instance has initiated the auto-rejoin procedure (in the **COUNT\_STAR** field). The [**events\_stages\_history\_long**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-history-long-table) table shows the time each of these auto-rejoin procedures was completed (in the **TIMER\_END** field).

#### 18.7.6.4 Exit Action

The [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action) system variable, which is available from MySQL 8.0.12 and MySQL 5.7.24, specifies what Group Replication does when the member leaves the group unintentionally due to an error or problem, and either fails to auto-rejoin or does not try. Note that in the case of an expelled member, the member does not know that it was expelled until it reconnects to the group, so the specified action is only taken if the member manages to reconnect, or if the member raises a suspicion on itself and expels itself.

In order of impact, the exit actions are as follows:

If **READ\_ONLY** is the exit action, the instance switches MySQL to super read only mode by setting the system variable [**super\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) to **ON**. When the member is in super read only mode, clients cannot make any updates, even if they have the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege). However, clients can still read data, and because updates are no longer being made, there is a probability of stale reads which increases over time. With this setting, you therefore need to pro-actively monitor the servers for failures. This exit action is the default from MySQL 8.0.15. After this exit action is taken, the member's status is displayed as **ERROR** in the view of the group.

If **OFFLINE\_MODE** is the exit action, the instance switches MySQL to offline mode by setting the system variable [**offline\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_offline_mode) to **ON**. When the member is in offline mode, connected client users are disconnected on their next request and connections are no longer accepted, with the exception of client users that have the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege). Group Replication also sets the system variable [**super\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) to **ON**, so clients cannot make any updates, even if they have connected with the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) or [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege. This exit action prevents both updates and stale reads (with the exception of reads by client users with the stated privileges), and enables proxy tools such as MySQL Router to recognize that the server is unavailable and redirect client connections. It also leaves the instance running so that an administrator can attempt to resolve the issue without shutting down MySQL. This exit action is available from MySQL 8.0.18. After this exit action is taken, the member's status is displayed as **ERROR** in the view of the group (not **OFFLINE**, which means a member has Group Replication functionality available but does not currently belong to a group).

If **ABORT\_SERVER** is the exit action, the instance shuts down MySQL. Instructing the member to shut itself down prevents all stale reads and client updates, but it means that the MySQL Server instance is unavailable and must be restarted, even if the issue could have been resolved without that step. This exit action was the default from MySQL 8.0.12, when the system variable was added, to MySQL 8.0.15 inclusive. After this exit action is taken, the member is removed from the listing of servers in the view of the group.

Bear in mind that operator intervention is required whatever exit action is set, as an ex-member that has exhausted its auto-rejoin attempts (or never had any) and has been expelled from the group is not allowed to rejoin without a restart of Group Replication. The exit action only influences whether or not clients can still read data on the server that was unable to rejoin the group, and whether or not the server stays running.

**Important**

If a failure occurs before the member has successfully joined the group, the exit action specified by [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action) is not taken. This is the case if there is a failure during the local configuration check, or a mismatch between the configuration of the joining member and the configuration of the group. In these situations, the [**super\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) system variable is left with its original value, and the server does not shut down MySQL. To ensure that the server cannot accept updates when Group Replication did not start, we therefore recommend that [**super\_read\_only=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) is set in the server's configuration file at startup, which Group Replication changes to **OFF** on primary members after it has been started successfully. This safeguard is particularly important when the server is configured to start Group Replication on server boot ([**group\_replication\_start\_on\_boot=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot)), but it is also useful when Group Replication is started manually using a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) command.

If a failure occurs after the member has successfully joined the group, the specified exit action is taken. This is the case in the following situations:

Applier error - There is an error in the replication applier. This issue is not recoverable.

Distributed recovery not possible - There is an issue that means Group Replication's distributed recovery process (which uses remote cloning operations and state transfer from the binary log) cannot be completed. Group Replication retries distributed recovery automatically where this makes sense, but stops if there are no more options to complete the process. For details, see [Section 18.5.3.4, “Fault Tolerance for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-fault).

Group configuration change error - An error occurred during a group-wide configuration change carried out using a UDF, as described in [Section 18.5.1, “Configuring an Online Group”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-configuring-online-group).

Primary election error - An error occurred during election of a new primary member for a group in single-primary mode, as described in [Section 18.1.3.1, “Single-Primary Mode”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-single-primary-mode).

Unreachable majority timeout - The member has lost contact with a majority of the group members so is in a minority, and a timeout that was set by the [**group\_replication\_unreachable\_majority\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_unreachable_majority_timeout) system variable has expired.

Member expelled from group - A suspicion has been raised on the member, and any timeout set by the [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) system variable has expired, and the member has resumed communication with the group and found that it has been expelled.

Out of auto-rejoin attempts - The [**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries) system variable was set to specify a number of auto-rejoin attempts after a loss of majority or expulsion, and the member completed this number of attempts without success.

The following table summarizes the failure scenarios and actions in each case:

**Table 18.4 Exit actions in Group Replication failure situations**

| **Failure situation** | **Group Replication started with START GROUP\_REPLICATION** | **Group Replication started with group\_replication\_start\_on\_boot =ON** |
| --- | --- | --- |
| **Member fails local configuration check**  **Mismatch between joining member and group configuration** | **super\_read\_only** and **offline\_mode** unchanged  MySQL continues running  Set **super\_read\_only=ON** at startup to prevent updates | **super\_read\_only** and **offline\_mode** unchanged  MySQL continues running  Set **super\_read\_only=ON** at startup to prevent updates (Important) |
| **Applier error on member**  **Distributed recovery not possible**  **Group configuration change error**  **Primary election error**  **Unreachable majority timeout**  **Member expelled from group**  **Out of auto-rejoin attempts** | **super\_read\_only** set to **ON**  OR  **offline\_mode** and **super\_read\_only** set to **ON**  OR  MySQL shuts down | **super\_read\_only** set to **ON**  OR  **offline\_mode** and **super\_read\_only** set to **ON**  OR  MySQL shuts down |

## 18.8 Upgrading Group Replication

[18.8.1 Combining Different Member Versions in a Group](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-online-upgrade-combining-versions)

[18.8.2 Group Replication Offline Upgrade](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-offline-upgrade)

[18.8.3 Group Replication Online Upgrade](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-online-upgrade)

This section explains how to upgrade a Group Replication setup. The basic process of upgrading members of a group is the same as upgrading stand-alone instances, see [Section 2.11, “Upgrading MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading) for the actual process of doing upgrade and types available. Choosing between an in-place or logical upgrade depends on the amount of data stored in the group. Usually an in-place upgrade is faster, and therefore is recommended. You should also consult [Section 17.5.3, “Upgrading a Replication Setup”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-upgrade).

While you are in the process of upgrading an online group, in order to maximize availability, you might need to have members with different MySQL Server versions running at the same time. Group Replication includes compatibility policies that enable you to safely combine members running different versions of MySQL in the same group during the upgrade procedure. Depending on your group, the effects of these policies might affect the order in which you should upgrade group members. For details, see [Section 18.8.1, “Combining Different Member Versions in a Group”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-online-upgrade-combining-versions).

If your group can be taken fully offline see [Section 18.8.2, “Group Replication Offline Upgrade”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-offline-upgrade). If your group needs to remain online, as is common with production deployments, see [Section 18.8.3, “Group Replication Online Upgrade”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-online-upgrade) for the different approaches available for upgrading a group with minimal downtime.

### 18.8.1 Combining Different Member Versions in a Group

[18.8.1.1 Member Versions During Upgrades](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-compatibility-upgrade)

[18.8.1.2 Group Replication Communication Protocol Version](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-compatibility-communication)

Group Replication is versioned according to the MySQL Server version that the Group Replication plugin was bundled with. For example, if a member is running MySQL 5.7.26 then that is the version of the Group Replication plugin. To check the version of MySQL Server on a group member issue:

**SELECT MEMBER\_HOST,MEMBER\_PORT,MEMBER\_VERSION FROM performance\_schema.replication\_group\_members;**

+-------------+-------------+----------------+

| member\_host | member\_port | member\_version |

+-------------+-------------+----------------+

| example.com | 3306 | 8.0.13 |

+-------------+-------------+----------------+

For guidance on understanding the MySQL Server version and selecting a version, see [Section 2.1.2, “Which MySQL Version and Distribution to Install”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#which-version).

For optimal compatibility and performance, all members of a group should run the same version of MySQL Server and therefore of Group Replication. However, while you are in the process of upgrading an online group, in order to maximize availability, you might need to have members with different MySQL Server versions running at the same time. Depending on the changes made between the versions of MySQL, you could encounter incompatibilities in this situation. For example, if a feature has been deprecated between major versions, then combining the versions in a group might cause members that rely on the deprecated feature to fail. Conversely, writing to a member running a newer MySQL version while there are read-write members in the group running an older MySQL version might cause issues on members that lack functions introduced in the newer release.

To prevent these issues, Group Replication includes compatibility policies that enable you to safely combine members running different versions of MySQL in the same group. A member applies these policies to decide whether to join the group normally, or join in read-only mode, or not join the group, depending on which choice results in the safe operation of the joining member and of the existing members of the group. In an upgrade scenario, each server must leave the group, be upgraded, and rejoin the group with its new server version. At this point the member applies the policies for its new server version, which might have changed from the policies it applied when it originally joined the group.

As the administrator, you can instruct any server to attempt to join any group by configuring the server appropriately and issuing a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement. A decision to join or not join the group, or to join the group in read-only mode, is made and implemented by the joining member itself after you attempt to add it to the group. The joining member receives information on the MySQL Server versions of the current group members, assesses its own compatibility with those members, and applies the policies used in its own MySQL Server version (not the policies used by the existing members) to decide whether it is compatible.

The compatibility policies that a joining member applies when attempting to join a group are as follows:

A member does not join a group if it is running a lower MySQL Server version than the lowest version that the existing group members are running.

A member joins a group normally if it is running the same MySQL Server version as the lowest version that the existing group members are running.

A member joins a group but remains in read-only mode if it is running a higher MySQL Server version than the lowest version that the existing group members are running. This behavior only makes a difference when the group is running in multi-primary mode, because in a group that is running in single-primary mode, newly added members default to being read-only in any case.

Members running MySQL 8.0.17 or higher take into account the patch version of the release when checking their compatibility. Members running MySQL 8.0.16 or lower, or MySQL 5.7, only take into account the major version. For example, if you have a group with members all running MySQL version 8.0.13:

A member that is running MySQL version 5.7 does not join.

A member running MySQL 8.0.16 joins normally (because it considers the major version).

A member running MySQL 8.0.17 joins but remains in read-only mode (because it considers the patch version).

Note that joining members running releases before MySQL 5.7.27 check against all group members to find whether their own MySQL Server major version is lower. They therefore fail this check for a group where any members are running MySQL 8.0 releases, and cannot join the group even if it already has other members running MySQL 5.7. From MySQL 5.7.27, joining members only check against the group members that are running the lowest major version, so they can join a mixed version group where other MySQL 5.7 servers are present.

In a multi-primary mode group with members that use different MySQL Server versions, Group Replication automatically manages the read-write and read-only status of members running MySQL 8.0.17 or higher. If a member leaves the group, the members running the version that is now the lowest are automatically set to read-write mode. When you change a group that was running in single-primary mode to run in multi-primary mode, using the [**group\_replication\_switch\_to\_multi\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-multi-primary-mode) UDF, Group Replication automatically sets members to the correct mode. Members are automatically placed in read-only mode if they are running a higher MySQL server version than the lowest version present in the group, and members running the lowest version are placed in read-write mode.

#### 18.8.1.1 Member Versions During Upgrades

During an online upgrade procedure, if the group is in single-primary mode, all the servers that are not currently offline for upgrading function as they did before. The group elects a new primary whenever necessary, following the election policies described in [Section 18.1.3.1, “Single-Primary Mode”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-single-primary-mode). Note that if you require the primary to remain the same throughout (except when it is being upgraded itself), you must first upgrade all of the secondaries to a version higher than or equal to the target primary member version, then upgrade the primary last. The primary cannot remain as the primary unless it is running the lowest MySQL Server version in the group. After the primary has been upgraded, you can use the [**group\_replication\_set\_as\_primary()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-as-primary) UDF to reappoint it as the primary.

If the group is in multi-primary mode, fewer online members are available to perform writes during the upgrade procedure, because upgraded members join in read-only mode after their upgrade. From MySQL 8.0.17, this applies to upgrades between patch versions, and for lower releases, this only applies to upgrades between major versions. When all members have been upgraded to the same release, from MySQL 8.0.17, they all change back to read-write mode automatically. For earlier releases, you must set [**super\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) to **OFF** manually on each member that should function as a primary following the upgrade.

To deal with a problem situation, for example if you have to roll back an upgrade or add extra capacity to a group in an emergency, it is possible to allow a member to join an online group although it is running a lower MySQL Server version than the lowest version in use by other group members. The Group Replication system variable [**group\_replication\_allow\_local\_lower\_version\_join**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_allow_local_lower_version_join) can be used in such situations to override the normal compatibility policies. It is important to note that setting the option to **ON** does not make the new member compatible with the group, and allows it to join the group without any safeguards against incompatible behaviors by the existing members. The option must therefore only be used carefully in specific situations, and you must take additional precautions to avoid the new member failing due to normal group activity. For details of these precautions, see the description for [**group\_replication\_allow\_local\_lower\_version\_join**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_allow_local_lower_version_join).

#### 18.8.1.2 Group Replication Communication Protocol Version

A replication group uses a Group Replication communication protocol version that can differ from the MySQL Server version of the members. To check the group's communication protocol version, issue the following statement on any member:

**SELECT group\_replication\_get\_communication\_protocol();**

The return value shows the oldest MySQL Server version that can join this group and use the group's communication protocol. Versions from MySQL 5.7.14 allow compression of messages, and versions from MySQL 8.0.16 also allow fragmentation of messages. Note that the [**group\_replication\_get\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-get-communication-protocol) UDF returns the minimum MySQL version that the group supports, which might differ from the version number that was passed to the [**group\_replication\_set\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-communication-protocol) UDF, and from the MySQL Server version that is installed on the member where you use the UDF.

When you upgrade all the members of a replication group to a new MySQL Server release, the Group Replication communication protocol version is not automatically upgraded, in case there is still a requirement to allow members at earlier releases to join. If you do not need to support older members and want to allow the upgraded members to use any added communication capabilities, after the upgrade use the [**group\_replication\_set\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-communication-protocol) UDF to upgrade the communication protocol, specifying the new MySQL Server version to which you have upgraded the members. For more information, see [Section 18.5.1.4, “Setting a Group's Communication Protocol Version”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-communication-protocol).

### 18.8.2 Group Replication Offline Upgrade

To perform an offline upgrade of a Group Replication group, you remove each member from the group, perform an upgrade of the member and then restart the group as usual. In a multi-primary group you can shutdown the members in any order. In a single-primary group, shutdown each secondary first and then finally the primary. See [Section 18.8.3.2, “Upgrading a Group Replication Member”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-upgrading-member) for how to remove members from a group and shutdown MySQL.

Once the group is offline, upgrade all of the members. See [Section 2.11, “Upgrading MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading) for how to perform an upgrade. When all members have been upgraded, restart the members.

If you upgrade all the members of a replication group when they are offline and then restart the group, the members join using the new release's Group Replication communication protocol version, so that becomes the group's communication protocol version. If you have a requirement to allow members at earlier releases to join, you can use the [**group\_replication\_set\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-communication-protocol) UDF to downgrade the communication protocol version, specifying the MySQL Server version of the prospective group member that has the oldest installed server version.

### 18.8.3 Group Replication Online Upgrade

[18.8.3.1 Online Upgrade Considerations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-online-upgrade-considerations)

[18.8.3.2 Upgrading a Group Replication Member](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-upgrading-member)

[18.8.3.3 Group Replication Online Upgrade Methods](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-online-upgrade-methods)

[18.8.3.4 Group Replication Upgrade with **mysqlbackup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-upgrade-with-mysqlbackup)

When you have a group running which you want to upgrade but you need to keep the group online to serve your application, you need to consider your approach to the upgrade. This section describes the different elements involved in an online upgrade, and various methods of how to upgrade your group.

#### 18.8.3.1 Online Upgrade Considerations

When upgrading an online group you should consider the following points:

Regardless of the way which you upgrade your group, it is important to disable any writes to group members until they are ready to rejoin the group.

When a member is stopped, the [**super\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) variable is set to on automatically, but this change is not persisted.

When MySQL 5.7.22 or MySQL 8.0.11 tries to join a group running MySQL 5.7.21 or lower it fails to join the group because MySQL 5.7.21 does not send its value of [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names).

#### 18.8.3.2 Upgrading a Group Replication Member

This section explains the steps required for upgrading a member of a group. This procedure is part of the methods described at [Section 18.8.3.3, “Group Replication Online Upgrade Methods”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-online-upgrade-methods). The process of upgrading a member of a group is common to all methods and is explained first. The way which you join upgraded members can depend on which method you are following, and other factors such as whether the group is operating in single-primary or multi-primary mode. How you upgrade the server instance, using either the in-place or provision approach, does not impact on the methods described here.

The process of upgrading a member consists of removing it from the group, following your chosen method of upgrading the member, and then rejoining the upgraded member to a group. The recommended order of upgrading members in a single-primary group is to upgrade all secondaries, and then upgrade the primary last. If the primary is upgraded before a secondary, a new primary using the older MySQL version is chosen, but there is no need for this step.

To upgrade a member of a group:

Connect a client to the group member and issue [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication). Before proceeding, ensure that the member's status is **OFFLINE** by monitoring the [**replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table) table.

Disable Group Replication from starting up automatically so that you can safely connect to the member after upgrading and configure it without it rejoining the group by setting [**group\_replication\_start\_on\_boot=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot).

**Important**

If an upgraded member has [**group\_replication\_start\_on\_boot=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) then it could rejoin the group before you can perform the MySQL upgrade procedure and could result in issues. For example, if the upgrade fails and the server restarts again, then a possibly broken server could try to join the group.

Stop the member, for example using [**mysqladmin shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin) or the [**SHUTDOWN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#shutdown) statement. Any other members in the group continue running.

Upgrade the member, using the in-place or provisioning approach. See [Section 2.11, “Upgrading MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading) for details. When restarting the upgraded member, because [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) is set to 0, Group Replication does not start on the instance, and therefore it does not rejoin the group.

Once the MySQL upgrade procedure has been performed on the member, [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) must be set to 1 to ensure Group Replication starts correctly after restart. Restart the member.

Connect to the upgraded member and issue [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication). This rejoins the member to the group. The Group Replication metadata is in place on the upgraded server, therefore there is usually no need to reconfigure Group Replication. The server has to catch up with any transactions processed by the group while the server was offline. Once it has caught up with the group, it becomes an online member of the group.

**Note**

The longer it takes to upgrade a server, the more time that member is offline and therefore the more time it takes for the server to catch up when added back to the group.

When an upgraded member joins a group which has any member running an earlier MySQL Server version, the upgraded member joins with [**super\_read\_only=on**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only). This ensures that no writes are made to upgraded members until all members are running the newer version. In a multi-primary mode group, when the upgrade has been completed successfully and the group is ready to process transactions, members that are intended as writeable primaries must be set to read-write mode. From MySQL 8.0.17, when all members of a group have been upgraded to the same release, they all change back to read-write mode automatically. For earlier releases you must set each member manually to read-write mode. Connect to each member and issue:

**SET GLOBAL super\_read\_only=OFF;**

#### 18.8.3.3 Group Replication Online Upgrade Methods

Choose one of the following methods of upgrading a Group Replication group:

##### Rolling In-Group Upgrade

This method is supported provided that servers running a newer version are not generating workload to the group while there are still servers with an older version in it. In other words servers with a newer version can join the group only as secondaries. In this method there is only ever one group, and each server instance is removed from the group, upgraded and then rejoined to the group.

This method is well suited to single-primary groups. When the group is operating in single-primary mode, if you require the primary to remain the same throughout (except when it is being upgraded itself), it should be the last member to be upgraded. The primary cannot remain as the primary unless it is running the lowest MySQL Server version in the group. After the primary has been upgraded, you can use the [**group\_replication\_set\_as\_primary()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-as-primary) UDF to reappoint it as the primary. If you do not mind which member is the primary, the members can be upgraded in any order. The group elects a new primary whenever necessary from among the members running the lowest MySQL Server version, following the election policies described in [Section 18.1.3.1, “Single-Primary Mode”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-single-primary-mode).

For groups operating in multi-primary mode, during a rolling in-group upgrade the number of primaries is decreased, causing a reduction in write availability. This is because if a member joins a group when it is running a higher MySQL Server version than the lowest version that the existing group members are running, it automatically remains in read-only mode ([**super\_read\_only=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only)). Note that members running MySQL 8.0.17 or higher take into account the patch version of the release when checking this, but members running MySQL 8.0.16 or lower, or MySQL 5.7, only take into account the major version. When all members have been upgraded to the same release, from MySQL 8.0.17, they all change back to read-write mode automatically. For earlier releases, you must set [**super\_read\_only=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) manually on each member that should function as a primary following the upgrade.

For full information on version compatibility in a group and how this influences the behavior of a group during an upgrade process, see [Section 18.8.1, “Combining Different Member Versions in a Group”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-online-upgrade-combining-versions) .

##### Rolling Migration Upgrade

In this method you remove members from the group, upgrade them and then create a second group using the upgraded members. For groups operating in multi-primary mode, during this process the number of primaries is decreased, causing a reduction in write availability. This does not impact groups operating in single-primary mode.

Because the group running the older version is online while you are upgrading the members, you need the group running the newer version to catch up with any transactions executed while the members were being upgraded. Therefore one of the servers in the new group is configured as a replica of a primary from the older group. This ensures that the new group catches up with the older group. Because this method relies on an asynchronous replication channel which is used to replicate data from one group to another, it is supported under the same assumptions and requirements of asynchronous source-replica replication, see [Chapter 17, *Replication*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html). For groups operating in single-primary mode, the asynchronous replication connection to the old group must send data to the primary in the new group, for a multi-primary group the asynchronous replication channel can connect to any primary.

The process is to:

remove members from the original group running the older server version one by one, see [Section 18.8.3.2, “Upgrading a Group Replication Member”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-upgrading-member)

upgrade the server version running on the member, see [Section 2.11, “Upgrading MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading). You can either follow an in-place or provision approach to upgrading.

create a new group with the upgraded members, see [Chapter 18, *Group Replication*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html). In this case you need to configure a new group name on each member (because the old group is still running and using the old name), bootstrap an initial upgraded member, and then add the remaining upgraded members.

set up an asynchronous replication channel between the old group and the new group, see [Section 17.1.3.4, “Setting Up Replication Using GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-howto). Configure the older primary to function as the asynchronous replication source server and the new group member as a GTID-based replica.

Before you can redirect your application to the new group, you must ensure that the new group has a suitable number of members, for example so that the group can handle the failure of a member. Issue **SELECT \* FROM performance\_schema.replication\_group\_members** and compare the initial group size and the new group size. Wait until all data from the old group is propagated to the new group and then drop the asynchronous replication connection and upgrade any missing members.

##### Rolling Duplication Upgrade

In this method you create a second group consisting of members which are running the newer version, and the data missing from the older group is replicated to the newer group. This assumes that you have enough servers to run both groups simultaneously. Due to the fact that during this process the number of primaries is not decreased, for groups operating in multi-primary mode there is no reduction in write availability. This makes rolling duplication upgrade well suited to groups operating in multi-primary mode. This does not impact groups operating in single-primary mode.

Because the group running the older version is online while you are provisioning the members in the new group, you need the group running the newer version to catch up with any transactions executed while the members were being provisioned. Therefore one of the servers in the new group is configured as a replica of a primary from the older group. This ensures that the new group catches up with the older group. Because this method relies on an asynchronous replication channel which is used to replicate data from one group to another, it is supported under the same assumptions and requirements of asynchronous source-replica replication, see [Chapter 17, *Replication*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html). For groups operating in single-primary mode, the asynchronous replication connection to the old group must send data to the primary in the new group, for a multi-primary group the asynchronous replication channel can connect to any primary.

The process is to:

deploy a suitable number of members so that the group running the newer version can handle failure of a member

take a backup of the existing data from a member of the group

use the backup from the older member to provision the members of the new group, see [Section 18.8.3.4, “Group Replication Upgrade with **mysqlbackup**”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-upgrade-with-mysqlbackup) for one method.

**Note**

You must restore the backup to the same version of MySQL which the backup was taken from, and then perform an in-place upgrade. For instructions, see [Section 2.11, “Upgrading MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading).

create a new group with the upgraded members, see [Chapter 18, *Group Replication*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html). In this case you need to configure a new group name on each member (because the old group is still running and using the old name), bootstrap an initial upgraded member, and then add the remaining upgraded members.

set up an asynchronous replication channel between the old group and the new group, see [Section 17.1.3.4, “Setting Up Replication Using GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-howto). Configure the older primary to function as the asynchronous replication source server and the new group member as a GTID-based replica.

Once the ongoing data missing from the newer group is small enough to be quickly transferred, you must redirect write operations to the new group. Wait until all data from the old group is propagated to the new group and then drop the asynchronous replication connection.

#### 18.8.3.4 Group Replication Upgrade with ****mysqlbackup****

As part of a provisioning approach you can use MySQL Enterprise Backup to copy and restore the data from a group member to new members. However you cannot use this technique to directly restore a backup taken from a member running an older version of MySQL to a member running a newer version of MySQL. The solution is to restore the backup to a new server instance which is running the same version of MySQL as the member which the backup was taken from, and then upgrade the instance. This process consists of:

Take a backup from a member of the older group using **mysqlbackup**. See [Section 18.5.6, “Using MySQL Enterprise Backup with Group Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-enterprise-backup).

Deploy a new server instance, which must be running the same version of MySQL as the older member where the backup was taken.

Restore the backup from the older member to the new instance using **mysqlbackup**.

Upgrade MySQL on the new instance, see [Section 2.11, “Upgrading MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading).

Repeat this process to create a suitable number of new instances, for example to be able to handle a failover. Then join the instances to a group based on the [Section 18.8.3.3, “Group Replication Online Upgrade Methods”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-online-upgrade-methods).`

## 18.9 Group Replication System Variables

This section lists the system variables that are specific to the Group Replication plugin. Every configuration option is prefixed with "**group\_replication**".

Most system variables for Group Replication are described as dynamic, and their values can be changed while the server is running. However, in most cases, the change only takes effect after you stop and restart Group Replication on the group member using a [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) statement followed by a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement. Changes to the following system variables take effect without stopping and restarting Group Replication:

[**group\_replication\_advertise\_recovery\_endpoints**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_advertise_recovery_endpoints)

[**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries)

[**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency)

[**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action)

[**group\_replication\_flow\_control\_applier\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_applier_threshold)

[**group\_replication\_flow\_control\_certifier\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_certifier_threshold)

[**group\_replication\_flow\_control\_hold\_percent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_hold_percent)

[**group\_replication\_flow\_control\_max\_commit\_quota**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_max_commit_quota)

[**group\_replication\_flow\_control\_member\_quota\_percent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_member_quota_percent)

[**group\_replication\_flow\_control\_min\_quota**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_min_quota)

[**group\_replication\_flow\_control\_min\_recovery\_quota**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_min_recovery_quota)

[**group\_replication\_flow\_control\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_mode)

[**group\_replication\_flow\_control\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_period)

[**group\_replication\_flow\_control\_release\_percent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_release_percent)

[**group\_replication\_force\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_force_members)

[**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist)

[**group\_replication\_ip\_whitelist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_whitelist)

[**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout)

[**group\_replication\_member\_weight**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_weight)

[**group\_replication\_transaction\_size\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_transaction_size_limit)

[**group\_replication\_unreachable\_majority\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_unreachable_majority_timeout)

Most system variables for Group Replication can have different values on different group members. For the following system variables, it is advisable to set the same value on all members of a group in order to avoid unnecessary rollback of transactions, failure of message delivery, or failure of message recovery:

[**group\_replication\_auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_auto_increment_increment)

[**group\_replication\_communication\_max\_message\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_communication_max_message_size)

[**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold)

[**group\_replication\_message\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_message_cache_size)

[**group\_replication\_transaction\_size\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_transaction_size_limit)

Some system variables on a Group Replication group member, including some Group Replication-specific system variables and some general system variables, are group-wide configuration settings. These system variables must have the same value on all group members, cannot be changed while Group Replication is running, and require a full reboot of the group (a bootstrap by a server with [**group\_replication\_bootstrap\_group=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group)) in order for the value change to take effect. These conditions apply to the following system variables:

[**group\_replication\_single\_primary\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_single_primary_mode)

[**group\_replication\_enforce\_update\_everywhere\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_enforce_update_everywhere_checks)

[**group\_replication\_gtid\_assignment\_block\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_gtid_assignment_block_size)

[**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption)

[**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names)

[**transaction\_write\_set\_extraction**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_transaction_write_set_extraction)

From MySQL 8.0.16, you can use the [**group\_replication\_switch\_to\_single\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-single-primary-mode) and [**group\_replication\_switch\_to\_multi\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-multi-primary-mode) UDFs to change the values of [**group\_replication\_single\_primary\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_single_primary_mode) and [**group\_replication\_enforce\_update\_everywhere\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_enforce_update_everywhere_checks) while the group is still running. For more information, see [Section 18.5.1.2, “Changing a Group's Mode”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-changing-group-mode).

**Important**

A number of system variables for Group Replication are not completely validated during server startup if they are passed as command line arguments to the server. These system variables include [**group\_replication\_group\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_name), [**group\_replication\_single\_primary\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_single_primary_mode), [**group\_replication\_force\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_force_members), the SSL variables, and the flow control system variables. They are only fully validated after the server has started.

System variables for Group Replication that specify IP addresses or host names for group members are not validated until a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement is issued. Group Replication's Group Communication System (GCS) is not available to validate the values until that point.

The system variables that are specific to the Group Replication plugin are as follows:

**[group\_replication\_advertise\_recovery\_endpoints](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_advertise_recovery_endpoints)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-advertise-recovery-endpoints=value** |
| **Introduced** | 8.0.21 |
| **System Variable** | [**group\_replication\_advertise\_recovery\_endpoints**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_advertise_recovery_endpoints) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **DEFAULT** |

The value of this system variable can be changed while Group Replication is running. The change takes effect immediately on the member. However, a joining member that already received the previous value of the system variable continues to use that value. Only members that join after the value change receive the new value.

[**group\_replication\_advertise\_recovery\_endpoints**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_advertise_recovery_endpoints) specifies how a joining member can establish a connection to an existing member for state transfer for distributed recovery. The connection is used for both remote cloning operations and state transfer from the donor's binary log.

A value of **DEFAULT**, which is the default setting, means joining members use the existing member's standard SQL client connection, as specified by MySQL Server's [**hostname**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_hostname) and [**port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_port) system variables. If an alternative port number is specified by the [**report\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_port) system variable, that one is used instead. The Performance Schema table [**replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table) shows this connection's address and port number in the **MEMBER\_HOST** and **MEMBER\_PORT** fields. This is the behavior of group members at releases up to and including MySQL 8.0.20.

Instead of **DEFAULT**, you can specify one or more distributed recovery endpoints, which the existing member advertises to joining members for them to use. Offering distributed recovery endpoints lets administrators control distributed recovery traffic separately from regular MySQL client connections to the group members. Joining members try each of the endpoints in turn in the order they are specified on the list.

Specify the distributed recovery endpoints as a comma-separated list of IP addresses and port numbers, for example:

group\_replication\_advertise\_recovery\_endpoints= "127.0.0.1:3306,127.0.0.1:4567,[::1]:3306,localhost:3306"

IPv4 and IPv6 addresses and host names can be used in any combination. IPv6 addresses must be specified in square brackets. Host names must resolve to a local IP address. Wildcard address formats cannot be used, and you cannot specify an empty list. Note that the standard SQL client connection is not automatically included on a list of distributed recovery endpoints. If you want to use it as an endpoint, you must include it explicitly on the list.

For details of how to select IP addresses and ports as distributed recovery endpoints, and how joining members use them, see [Section 18.5.3.1.1, “Selecting addresses for distributed recovery endpoints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-connections-endpoints). A summary of the requirements is as follows:

The IP addresses do not have to be configured for MySQL Server, but they do have to be assigned to the server.

The ports do have to be configured for MySQL Server using the [**port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_port), [**report\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_port), or [**admin\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_admin_port) system variable.

Appropriate permissions are required for the replication user for distributed recovery if the [**admin\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_admin_port) is used.

The IP addresses do not need to be added to the Group Replication allowlist specified by the [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) or [**group\_replication\_ip\_whitelist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_whitelist) system variable.

The SSL requirements for the connection are as specified by the **group\_replication\_recovery\_ssl\_\*** options.

**[group\_replication\_allow\_local\_lower\_version\_join](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_allow_local_lower_version_join)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-allow-local-lower-version-join[={OFF|ON}]** |
| **System Variable** | [**group\_replication\_allow\_local\_lower\_version\_join**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_allow_local_lower_version_join) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_allow\_local\_lower\_version\_join**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_allow_local_lower_version_join) allows the current server to join the group even if it is running a lower MySQL Server version than the group. With the default setting **OFF**, servers are not permitted to join a replication group if they are running a lower version than the existing group members. This standard policy ensures that all members of a group are able to exchange messages and apply transactions. Note that members running MySQL 8.0.17 or higher take into account the patch version of the release when checking their compatibility. Members running MySQL 8.0.16 or lower, or MySQL 5.7, only take into account the major version.

Set [**group\_replication\_allow\_local\_lower\_version\_join**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_allow_local_lower_version_join) to **ON** only in the following scenarios:

A server must be added to the group in an emergency in order to improve the group's fault tolerance, and only older versions are available.

You want to roll back an upgrade for one or more replication group members without shutting down the whole group and bootstrapping it again.

**Warning**

Setting this option to **ON** does not make the new member compatible with the group, and allows it to join the group without any safeguards against incompatible behaviors by the existing members. To ensure the new member's correct operation, take both of the following precautions:

Before the server running the lower version joins the group, stop all writes on that server.

From the point where the server running the lower version joins the group, stop all writes on the other servers in the group.

Without these precautions, the server running the lower version is likely to experience difficulties and terminate with an error.

**[group\_replication\_auto\_increment\_increment](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_auto_increment_increment)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-auto-increment-increment=#** |
| **System Variable** | [**group\_replication\_auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_auto_increment_increment) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **7** |
| **Minimum Value** | **1** |
| **Maximum Value** | **65535** |

This system variable should have the same value on all group members. You cannot change the value of this system variable while Group Replication is running. You must stop Group Replication, change the value of the system variable, then restart Group Replication, on each of the group members. During this process, the value of the system variable is permitted to differ between group members, but some transactions on group members might be rolled back.

[**group\_replication\_auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_auto_increment_increment) determines the interval between successive values for auto-incremented columns for transactions that execute on this server instance. Adding an interval avoids the selection of duplicate auto-increment values for writes on group members, which causes rollback of transactions. The default value of 7 represents a balance between the number of usable values and the permitted maximum size of a replication group (9 members). If your group has more or fewer members, you can set this system variable to match the expected number of group members before Group Replication is started.

When Group Replication is started on a server instance, the value of the server system variable [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) is changed to this value, and the value of the server system variable [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) is changed to the server ID. The changes are reverted when Group Replication is stopped. These changes are only made and reverted if [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) and [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) each have their default value of 1. If their values have already been modified from the default, Group Replication does not alter them. From MySQL 8.0, the system variables are also not modified when Group Replication is in single-primary mode, where only one server writes.

**[group\_replication\_autorejoin\_tries](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_autorejoin_tries)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-autorejoin-tries=#** |
| **Introduced** | 8.0.16 |
| **System Variable** | [**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value (≥ 8.0.21)** | **3** |
| **Default Value (≤ 8.0.20)** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **2016** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately. The system variable's current value is read when an issue occurs that means the behavior is needed.

[**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries) specifies the number of tries that a member makes to automatically rejoin the group if it is expelled, or if it is unable to contact a majority of the group before the [**group\_replication\_unreachable\_majority\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_unreachable_majority_timeout) setting is reached. When the member's expulsion or unreachable majority timeout is reached, it makes an attempt to rejoin (using the current plugin option values), then continues to make further auto-rejoin attempts up to the specified number of tries. After an unsuccessful auto-rejoin attempt, the member waits 5 minutes before the next try. If the specified number of tries is exhausted without the member rejoining or being stopped, the member proceeds to the action specified by the [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action) system variable.

Up to MySQL 8.0.20, the default setting is 0, meaning that the member does not try to rejoin automatically. From MySQL 8.0.21, the default setting is 3, meaning that the member automatically makes 3 attempts to rejoin the group, with 5 minutes between each. You can specify a maximum of 2016 tries.

During and between auto-rejoin attempts, a member remains in super read only mode and does not accept writes, but reads can still be made on the member, with an increasing likelihood of stale reads over time. If you cannot tolerate the possibility of stale reads for any period of time, set [**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries) to 0. For more information on the auto-rejoin feature, and considerations when choosing a value for this option, see [Section 18.7.6.3, “Auto-Rejoin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-responses-failure-rejoin).

**[group\_replication\_bootstrap\_group](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_bootstrap_group)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-bootstrap-group[={OFF|ON}]** |
| **System Variable** | [**group\_replication\_bootstrap\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

[**group\_replication\_bootstrap\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group) configures this server to bootstrap the group. This system variable must only be set on one server, and only when starting the group for the first time or restarting the entire group. After the group has been bootstrapped, set this option to **OFF**. It should be set to **OFF** both dynamically and in the configuration files. Starting two servers or restarting one server with this option set while the group is running may lead to an artificial split brain situation, where two independent groups with the same name are bootstrapped.

**[group\_replication\_clone\_threshold](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_clone_threshold)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-clone-threshold=#** |
| **Introduced** | 8.0.17 |
| **System Variable** | [**group\_replication\_clone\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_clone_threshold) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **9223372036854775807** |
| **Minimum Value** | **1** |
| **Maximum Value** | **9223372036854775807** |
| **Unit** | **transactions** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_clone\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_clone_threshold) specifies the transaction gap, as a number of transactions, between the existing member (donor) and the joining member (recipient) that triggers the use of a remote cloning operation for state transfer to the joining member during the distributed recovery process. If the transaction gap between the joining member and a suitable donor exceeds the threshold, Group Replication begins distributed recovery with a remote cloning operation. If the transaction gap is below the threshold, or if the remote cloning operation is not technically possible, Group Replication proceeds directly to state transfer from a donor's binary log.

**Warning**

Do not use a low setting for [**group\_replication\_clone\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_clone_threshold) in an active group. If a number of transactions above the threshold takes place in the group while the remote cloning operation is in progress, the joining member triggers a remote cloning operation again after restarting, and could continue this indefinitely. To avoid this situation, ensure that you set the threshold to a number higher than the number of transactions that you would expect to occur in the group during the time taken for the remote cloning operation.

To use this function, both the donor and the joining member must be set up beforehand to support cloning. For instructions, see [Section 18.5.3.2, “Cloning for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-cloning). When a remote cloning operation is carried out, Group Replication manages it for you, including the required server restart, provided that [**group\_replication\_start\_on\_boot=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) is set. If not, you must restart the server manually. The remote cloning operation replaces the existing data dictionary on the joining member, but Group Replication checks and does not proceed if the joining member has additional transactions that are not present on the other group members, because these transactions would be erased by the cloning operation.

The default setting (which is the maximum permitted sequence number for a transaction in a GTID) means that state transfer from a donor's binary log is virtually always attempted rather than cloning. However, note that Group Replication always attempts to execute a cloning operation, regardless of your threshold, if state transfer from a donor's binary log is impossible, for example because the transactions needed by the joining member are not available in the binary logs on any existing group member. If you do not want to use cloning at all in your replication group, do not install the clone plugin on the members.

**[group\_replication\_communication\_debug\_options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_communication_debug_options)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-communication-debug-options=value** |
| **System Variable** | [**group\_replication\_communication\_debug\_options**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_communication_debug_options) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **GCS\_DEBUG\_NONE** |
| **Valid Values** | **GCS\_DEBUG\_NONE**  **GCS\_DEBUG\_BASIC**  **GCS\_DEBUG\_TRACE**  **XCOM\_DEBUG\_BASIC**  **XCOM\_DEBUG\_TRACE**  **GCS\_DEBUG\_ALL** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_communication\_debug\_options**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_communication_debug_options) configures the level of debugging messages to provide for the different Group Replication components, such as the Group Communication System (GCS) and the group communication engine (XCom, a Paxos variant). The debug information is stored in the GCS\_DEBUG\_TRACE file in the data directory.

The set of available options, specified as strings, can be combined. The following options are available:

**GCS\_DEBUG\_NONE** disables all debugging levels for both GCS and XCom.

**GCS\_DEBUG\_BASIC** enables basic debugging information in GCS.

**GCS\_DEBUG\_TRACE** enables trace information in GCS.

**XCOM\_DEBUG\_BASIC** enables basic debugging information in XCom.

**XCOM\_DEBUG\_TRACE** enables trace information in XCom.

**GCS\_DEBUG\_ALL** enables all debugging levels for both GCS and XCom.

Setting the debug level to **GCS\_DEBUG\_NONE** only has an effect when provided without any other option. Setting the debug level to **GCS\_DEBUG\_ALL** overrides all other options.

**[group\_replication\_communication\_max\_message\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_communication_max_message_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-communication-max-message-size=#** |
| **Introduced** | 8.0.16 |
| **System Variable** | [**group\_replication\_communication\_max\_message\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_communication_max_message_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **10485760** |
| **Minimum Value** | **0** |
| **Maximum Value** | **1073741824** |
| **Unit** | **bytes** |

This system variable should have the same value on all group members. You cannot change the value of this system variable while Group Replication is running. You must stop Group Replication, change the value of the system variable, then restart Group Replication, on each of the group members. During this process, the value of the system variable is permitted to differ between group members, but some transactions on group members might be rolled back.

[**group\_replication\_communication\_max\_message\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_communication_max_message_size) specifies a maximum message size for Group Replication communications. Messages greater than this size are automatically split into fragments that are sent separately and reassembled by the recipients. For more information, see [Section 18.7.4, “Message Fragmentation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-performance-message-fragmentation).

A maximum message size of 10485760 bytes (10 MiB) is set by default, which means that fragmentation is used by default in releases from MySQL 8.0.16. The greatest permitted value is the same as the maximum value of the [**slave\_max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_max_allowed_packet) system variable, which is 1073741824 bytes (1 GB). The setting for [**group\_replication\_communication\_max\_message\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_communication_max_message_size) must be less than the [**slave\_max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_max_allowed_packet) setting, because the applier thread cannot handle message fragments larger than [**slave\_max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_max_allowed_packet). To switch off fragmentation, specify a zero value for [**group\_replication\_communication\_max\_message\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_communication_max_message_size).

In order for members of a replication group to use fragmentation, the group's communication protocol version must be MySQL 8.0.16 or above. Use the [**group\_replication\_get\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-get-communication-protocol) UDF to view the group's communication protocol version. If a lower version is in use, group members do not fragment messages. You can use the [**group\_replication\_set\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-communication-protocol) UDF to set the group's communication protocol to a higher version if all group members support it. For more information, see [Section 18.5.1.4, “Setting a Group's Communication Protocol Version”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-communication-protocol).

**[group\_replication\_components\_stop\_timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_components_stop_timeout)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-components-stop-timeout=#** |
| **System Variable** | [**group\_replication\_components\_stop\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_components_stop_timeout) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **31536000** |
| **Minimum Value** | **2** |
| **Maximum Value** | **31536000** |
| **Unit** | **seconds** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_components\_stop\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_components_stop_timeout) specifies the timeout, in seconds, that Group Replication waits for each of the components when shutting down.

**[group\_replication\_compression\_threshold](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_compression_threshold)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-compression-threshold=#** |
| **System Variable** | [**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1000000** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |
| **Unit** | **bytes** |

This system variable should have the same value on all group members. The value of this system variable can be changed while Group Replication is running. The change takes effect on each group member after you stop and restart Group Replication on the member. During this process, the value of the system variable is permitted to differ between group members, but message delivery does not have the same efficiency on all members.

[**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold) specifies the threshold value in bytes above which compression is applied to messages sent between group members. If this system variable is set to zero, compression is disabled.

Group Replication uses the LZ4 compression algorithm to compress messages sent in the group. Note that the maximum supported input size for the LZ4 compression algorithm is 2113929216 bytes. This limit is lower than the maximum possible value for the [**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold) system variable, which is matched to the maximum message size accepted by XCom. With the LZ4 compression algorithm, do not set a value greater than 2113929216 bytes for [**group\_replication\_compression\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_compression_threshold), because transactions above this size cannot be committed when message compression is enabled.

For more information, see [Section 18.7.3, “Message Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-message-compression).

**[group\_replication\_consistency](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_consistency)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-consistency=value** |
| **Introduced** | 8.0.14 |
| **System Variable** | [**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **EVENTUAL** |
| **Valid Values** | **EVENTUAL**  **BEFORE\_ON\_PRIMARY\_FAILOVER**  **BEFORE**  **AFTER**  **BEFORE\_AND\_AFTER** |

The value of this system variable can be changed while Group Replication is running. [**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) is a server system variable rather than a Group Replication plugin-specific variable, so a restart of Group Replication is not required for the change to take effect. Changing the session value of the system variable takes effect immediately, and changing the global value takes effect for new sessions that start after the change. The [**GROUP\_REPLICATION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_group-replication-admin) privilege is required to change the global setting for this system variable.

[**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) controls the transaction consistency guarantee which a group provides. You can configure the consistency globally or per transaction. [**group\_replication\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_consistency) also configures the fencing mechanism used by newly elected primaries in single primary groups. The effect of the variable must be considered for both read only (RO) and read write (RW) transactions. The following list shows the possible values of this variable, in order of increasing transaction consistency guarantee:

**EVENTUAL**

Both RO and RW transactions do not wait for preceding transactions to be applied before executing. This was the behavior of Group Replication before this variable was added. A RW transaction does not wait for other members to apply a transaction. This means that a transaction could be externalized on one member before the others. This also means that in the event of a primary failover, the new primary can accept new RO and RW transactions before the previous primary transactions are all applied. RO transactions could result in outdated values, RW transactions could result in a rollback due to conflicts.

**BEFORE\_ON\_PRIMARY\_FAILOVER**

New RO or RW transactions with a newly elected primary that is applying backlog from the old primary are held (not applied) until any backlog has been applied. This ensures that when a primary failover happens, intentionally or not, clients always see the latest value on the primary. This guarantees consistency, but means that clients must be able to handle the delay in the event that a backlog is being applied. Usually this delay should be minimal, but does depend on the size of the backlog.

**BEFORE**

A RW transaction waits for all preceding transactions to complete before being applied. A RO transaction waits for all preceding transactions to complete before being executed. This ensures that this transaction reads the latest value by only affecting the latency of the transaction. This reduces the overhead of synchronization on every RW transaction, by ensuring synchronization is used only on RO transactions. This consistency level also includes the consistency guarantees provided by **BEFORE\_ON\_PRIMARY\_FAILOVER**.

**AFTER**

A RW transaction waits until its changes have been applied to all of the other members. This value has no effect on RO transactions. This mode ensures that when a transaction is committed on the local member, any subsequent transaction reads the written value or a more recent value on any group member. Use this mode with a group that is used for predominantly RO operations to ensure that applied RW transactions are applied everywhere once they commit. This could be used by your application to ensure that subsequent reads fetch the latest data which includes the latest writes. This reduces the overhead of synchronization on every RO transaction, by ensuring synchronization is used only on RW transactions. This consistency level also includes the consistency guarantees provided by **BEFORE\_ON\_PRIMARY\_FAILOVER**.

**BEFORE\_AND\_AFTER**

A RW transaction waits for 1) all preceding transactions to complete before being applied and 2) until its changes have been applied on other members. A RO transaction waits for all preceding transactions to complete before execution takes place. This consistency level also includes the consistency guarantees provided by **BEFORE\_ON\_PRIMARY\_FAILOVER**.

For more information, see [Section 18.5.2, “Transaction Consistency Guarantees”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-consistency-guarantees).

**[group\_replication\_enforce\_update\_everywhere\_checks](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_enforce_update_everywhere_checks)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-enforce-update-everywhere-checks[={OFF|ON}]** |
| **System Variable** | [**group\_replication\_enforce\_update\_everywhere\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_enforce_update_everywhere_checks) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

This system variable is a group-wide configuration setting. It must have the same value on all group members, cannot be changed while Group Replication is running, and requires a full reboot of the group (a bootstrap by a server with [**group\_replication\_bootstrap\_group=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group)) in order for the value change to take effect. From MySQL 8.0.16, you can use the [**group\_replication\_switch\_to\_single\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-single-primary-mode) and [**group\_replication\_switch\_to\_multi\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-multi-primary-mode) UDFs to change the value of this system variable while the group is still running. For more information, see [Section 18.5.1.2, “Changing a Group's Mode”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-changing-group-mode).

[**group\_replication\_enforce\_update\_everywhere\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_enforce_update_everywhere_checks) enables or disables strict consistency checks for multi-primary update everywhere. The default is that checks are disabled. In single-primary mode, this option must be disabled on all group members. In multi-primary mode, when this option is enabled, statements are checked as follows to ensure they are compatible with multi-primary mode:

If a transaction is executed under the **SERIALIZABLE** isolation level, then its commit fails when synchronizing itself with the group.

If a transaction executes against a table that has foreign keys with cascading constraints, then the transaction fails to commit when synchronizing itself with the group.

**[group\_replication\_exit\_state\_action](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_exit_state_action)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-exit-state-action=value** |
| **Introduced** | 8.0.12 |
| **System Variable** | [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value (≥ 8.0.16)** | **READ\_ONLY** |
| **Default Value (≥ 8.0.12, ≤ 8.0.15)** | **ABORT\_SERVER** |
| **Valid Values (≥ 8.0.18)** | **ABORT\_SERVER**  **OFFLINE\_MODE**  **READ\_ONLY** |
| **Valid Values (≥ 8.0.12, ≤ 8.0.17)** | **ABORT\_SERVER**  **READ\_ONLY** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately. The system variable's current value is read when an issue occurs that means the behavior is needed.

[**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action) configures how Group Replication behaves when this server instance leaves the group unintentionally, for example after encountering an applier error, or in the case of a loss of majority, or when another member of the group expels it due to a suspicion timing out. The timeout period for a member to leave the group in the case of a loss of majority is set by the [**group\_replication\_unreachable\_majority\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_unreachable_majority_timeout) system variable, and the timeout period for suspicions is set by the [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) system variable. Note that an expelled group member does not know that it was expelled until it reconnects to the group, so the specified action is only taken if the member manages to reconnect, or if the member raises a suspicion on itself and expels itself.

When a group member is expelled due to a suspicion timing out or a loss of majority, if the member has the [**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries) system variable set to specify a number of auto-rejoin attempts, it first makes the specified number of attempts while in super read only mode, and then follows the action specified by [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action). Auto-rejoin attempts are not made in case of an applier error, because these are not recoverable.

When [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action) is set to **READ\_ONLY**, if the member exits the group unintentionally or exhausts its auto-rejoin attempts, the instance switches MySQL to super read only mode (by setting the system variable [**super\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) to **ON**). The **READ\_ONLY** exit action was the behavior for MySQL 8.0 releases before the system variable was introduced, and became the default again from MySQL 8.0.16.

When [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action) is set to **OFFLINE\_MODE**, if the member exits the group unintentionally or exhausts its auto-rejoin attempts, the instance switches MySQL to offline mode (by setting the system variable [**offline\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_offline_mode) to **ON**). In this mode, connected client users are disconnected on their next request and connections are no longer accepted, with the exception of client users that have the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege). Group Replication also sets the system variable [**super\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) to **ON**, so clients cannot make any updates, even if they have connected with the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) or [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege. The **OFFLINE\_MODE** exit action is available from MySQL 8.0.18.

When [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action) is set to **ABORT\_SERVER**, if the member exits the group unintentionally or exhausts its auto-rejoin attempts, the instance shuts down MySQL. This setting was the default from MySQL 8.0.12, when the system variable was added, to MySQL 8.0.15 inclusive.

**Important**

If a failure occurs before the member has successfully joined the group, the specified exit action is not taken. This is the case if there is a failure during the local configuration check, or a mismatch between the configuration of the joining member and the configuration of the group. In these situations, the [**super\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) system variable is left with its original value, connections continue to be accepted, and the server does not shut down MySQL. To ensure that the server cannot accept updates when Group Replication did not start, we therefore recommend that [**super\_read\_only=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) is set in the server's configuration file at startup, which Group Replication changes to **OFF** on primary members after it has been started successfully. This safeguard is particularly important when the server is configured to start Group Replication on server boot ([**group\_replication\_start\_on\_boot=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot)), but it is also useful when Group Replication is started manually using a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) command.

For more information on using this option, and the full list of situations in which the exit action is taken, see [Section 18.7.6.4, “Exit Action”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-responses-failure-exit).

**[group\_replication\_flow\_control\_applier\_threshold](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_flow_control_applier_threshold)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-flow-control-applier-threshold=#** |
| **System Variable** | [**group\_replication\_flow\_control\_applier\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_applier_threshold) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **25000** |
| **Minimum Value** | **0** |
| **Maximum Value** | **2147483647** |
| **Unit** | **transactions** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately.

[**group\_replication\_flow\_control\_applier\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_applier_threshold) specifies the number of waiting transactions in the applier queue that trigger flow control.

**[group\_replication\_flow\_control\_certifier\_threshold](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_flow_control_certifier_threshold)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-flow-control-certifier-threshold=#** |
| **System Variable** | [**group\_replication\_flow\_control\_certifier\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_certifier_threshold) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **25000** |
| **Minimum Value** | **0** |
| **Maximum Value** | **2147483647** |
| **Unit** | **transactions** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately.

[**group\_replication\_flow\_control\_certifier\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_certifier_threshold) specifies the number of waiting transactions in the certifier queue that trigger flow control.

**[group\_replication\_flow\_control\_hold\_percent](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_flow_control_hold_percent)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-flow-control-hold-percent=#** |
| **System Variable** | [**group\_replication\_flow\_control\_hold\_percent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_hold_percent) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **10** |
| **Minimum Value** | **0** |
| **Maximum Value** | **100** |
| **Unit** | **percentage** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately.

[**group\_replication\_flow\_control\_hold\_percent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_hold_percent) defines what percentage of the group quota remains unused to allow a cluster under flow control to catch up on backlog. A value of 0 implies that no part of the quota is reserved for catching up on the work backlog.

**[group\_replication\_flow\_control\_max\_commit\_quota](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_flow_control_max_commit_quota)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-flow-control-max-commit-quota=#** |
| **System Variable** | [**group\_replication\_flow\_control\_max\_commit\_quota**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_max_commit_quota) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **2147483647** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately.

[**group\_replication\_flow\_control\_max\_commit\_quota**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_max_commit_quota) defines the maximum flow control quota of the group, or the maximum available quota for any period while flow control is enabled. A value of 0 implies that there is no maximum quota set. The value of this system variable cannot be smaller than [**group\_replication\_flow\_control\_min\_quota**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_min_quota) and **group\_replication\_flow\_control\_min\_recovery\_quota**.

**[group\_replication\_flow\_control\_member\_quota\_percent](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_flow_control_member_quota_percent)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-flow-control-member-quota-percent=#** |
| **System Variable** | [**group\_replication\_flow\_control\_member\_quota\_percent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_member_quota_percent) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **100** |
| **Unit** | **percentage** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately.

[**group\_replication\_flow\_control\_member\_quota\_percent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_member_quota_percent) defines the percentage of the quota that a member should assume is available for itself when calculating the quotas. A value of 0 implies that the quota should be split equally between members that were writers in the last period.

**[group\_replication\_flow\_control\_min\_quota](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_flow_control_min_quota)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-flow-control-min-quota=#** |
| **System Variable** | [**group\_replication\_flow\_control\_min\_quota**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_min_quota) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **2147483647** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately.

[**group\_replication\_flow\_control\_min\_quota**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_min_quota) controls the lowest flow control quota that can be assigned to a member, independently of the calculated minimum quota executed in the last period. A value of 0 implies that there is no minimum quota. The value of this system variable cannot be larger than [**group\_replication\_flow\_control\_max\_commit\_quota**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_max_commit_quota).

**[group\_replication\_flow\_control\_min\_recovery\_quota](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_flow_control_min_recovery_quota)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-flow-control-min-recovery-quota=#** |
| **System Variable** | [**group\_replication\_flow\_control\_min\_recovery\_quota**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_min_recovery_quota) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **2147483647** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately.

[**group\_replication\_flow\_control\_min\_recovery\_quota**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_min_recovery_quota) controls the lowest quota that can be assigned to a member because of another recovering member in the group, independently of the calculated minimum quota executed in the last period. A value of 0 implies that there is no minimum quota. The value of this system variable cannot be larger than **group\_replication\_flow\_control\_max\_commit\_quota**.

**[group\_replication\_flow\_control\_mode](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_flow_control_mode)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-flow-control-mode=value** |
| **System Variable** | [**group\_replication\_flow\_control\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_mode) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **QUOTA** |
| **Valid Values** | **DISABLED**  **QUOTA** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately.

[**group\_replication\_flow\_control\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_mode) specifies the mode used for flow control.

**[group\_replication\_flow\_control\_period](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_flow_control_period)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-flow-control-period=#** |
| **System Variable** | [**group\_replication\_flow\_control\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_period) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1** |
| **Minimum Value** | **1** |
| **Maximum Value** | **60** |
| **Unit** | **seconds** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately.

[**group\_replication\_flow\_control\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_period) defines how many seconds to wait between flow control iterations, in which flow control messages are sent and flow control management tasks are run.

**[group\_replication\_flow\_control\_release\_percent](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_flow_control_release_percent)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-flow-control-release-percent=#** |
| **System Variable** | [**group\_replication\_flow\_control\_release\_percent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_release_percent) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **50** |
| **Minimum Value** | **0** |
| **Maximum Value** | **1000** |
| **Unit** | **percentage** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately.

[**group\_replication\_flow\_control\_release\_percent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_flow_control_release_percent) defines how the group quota should be released when flow control no longer needs to throttle the writer members, with this percentage being the quota increase per flow control period. A value of 0 implies that once the flow control thresholds are within limits the quota is released in a single flow control iteration. The range allows the quota to be released at up to 10 times current quota, as that allows a greater degree of adaptation, mainly when the flow control period is large and the quotas are very small.

**[group\_replication\_force\_members](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_force_members)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-force-members=value** |
| **System Variable** | [**group\_replication\_force\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_force_members) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

This system variable is used to force a new group membership. The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately. You only need to set the value of the system variable on one of the group members that is to remain in the group. For details of the situation in which you might need to force a new group membership, and a procedure to follow when using this system variable, see [Section 18.5.4, “Network Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-network-partitioning).

[**group\_replication\_force\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_force_members) specifies a list of peer addresses as a comma separated list, such as **host1:port1**,**host2:port2**. Any existing members that are not included in the list do not receive a new view of the group and are blocked. For each existing member that is to continue as a member, you must include the IP address or host name and the port, as they are given in the [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) system variable for each member. An IPv6 address must be specified in square brackets. For example:

"198.51.100.44:33061,[2001:db8:85a3:8d3:1319:8a2e:370:7348]:33061,example.org:33061"

The group communication engine for Group Replication (XCom) checks that the supplied IP addresses are in a valid format, and checks that you have not included any group members that are currently unreachable. Otherwise, the new configuration is not validated, so you must be careful to include only online servers that are reachable members of the group. Any incorrect values or invalid host names in the list could cause the group to be blocked with an invalid configuration.

It is important before forcing a new membership configuration to ensure that the servers to be excluded have been shut down. If they are not, shut them down before proceeding. Group members that are still online can automatically form new configurations, and if this has already taken place, forcing a further new configuration could create an artificial split-brain situation for the group.

After you have used the [**group\_replication\_force\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_force_members) system variable to successfully force a new group membership and unblock the group, ensure that you clear the system variable. [**group\_replication\_force\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_force_members) must be empty in order to issue a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement.

**[group\_replication\_group\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_group_name)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-group-name=value** |
| **System Variable** | [**group\_replication\_group\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_name) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

The value of this system variable cannot be changed while Group Replication is running.

[**group\_replication\_group\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_name) specifies the name of the group which this server instance belongs to, which must be a valid UUID. This UUID is used internally when setting GTIDs for Group Replication transactions in the binary log.

**Important**

A unique UUID must be used.

**[group\_replication\_group\_seeds](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_group_seeds)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-group-seeds=value** |
| **System Variable** | [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) is a list of group members to which a joining member can connect to obtain details of all the current group members. The joining member uses these details to select and connect to a group member to obtain the data needed for synchrony with the group. The list consists of a single internal network address or host name for each included seed member, as configured in the seed member's [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) system variable (not the seed member's SQL client connection, as specified by MySQL Server's [**hostname**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_hostname) and [**port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_port) system variables). The addresses of the seed members are specified as a comma separated list, such as **host1:port1**,**host2:port2**. An IPv6 address must be specified in square brackets. For example:

group\_replication\_group\_seeds= "198.51.100.44:33061,[2001:db8:85a3:8d3:1319:8a2e:370:7348]:33061, example.org:33061"

Note that the value you specify for this variable is not validated until a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement is issued and the Group Communication System (GCS) is available.

Usually this list consists of all members of the group, but you can choose a subset of the group members to be seeds. The list must contain at least one valid member address. Each address is validated when starting Group Replication. If the list does not contain any valid member addresses, issuing [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) fails.

When a server is joining a replication group, it attempts to connect to the first seed member listed in its [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) system variable. If the connection is refused, the joining member tries to connect to each of the other seed members in the list in order. If the joining member connects to a seed member but does not get added to the replication group as a result (for example, because the seed member does not have the joining member's address in its allowlist and closes the connection), the joining member continues to try the remaining seed members in the list in order.

A joining member must communicate with the seed member using the same protocol (IPv4 or IPv6) that the seed member advertises in the [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) option. For the purpose of IP address permissions for Group Replication, the allowlist on the seed member must include an IP address for the joining member for the protocol offered by the seed member, or a host name that resolves to an address for that protocol. This address or host name must be set up and permitted in addition to the joining member's [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) if the protocol for that address does not match the seed member's advertised protocol. If a joining member does not have a permitted address for the appropriate protocol, its connection attempt is refused. For more information, see [Section 18.6.1, “Group Replication IP Address Permissions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-ip-address-permissions).

**[group\_replication\_gtid\_assignment\_block\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_gtid_assignment_block_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-gtid-assignment-block-size=#** |
| **System Variable** | [**group\_replication\_gtid\_assignment\_block\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_gtid_assignment_block_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1000000** |
| **Minimum Value** | **1** |
| **Maximum Value (64-bit platforms)** | **9223372036854775807** |
| **Maximum Value (32-bit platforms)** | **4294967295** |

This system variable is a group-wide configuration setting. It must have the same value on all group members, cannot be changed while Group Replication is running, and requires a full reboot of the group (a bootstrap by a server with [**group\_replication\_bootstrap\_group=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group)) in order for the value change to take effect.

[**group\_replication\_gtid\_assignment\_block\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_gtid_assignment_block_size) specifies the number of consecutive GTIDs that are reserved for each group member. Each member consumes its own blocks and reserves more when needed.

**[group\_replication\_ip\_allowlist](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_ip_allowlist)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-ip-allowlist=value** |
| **Introduced** | 8.0.22 |
| **System Variable** | [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **AUTOMATIC** |

[**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) is available from MySQL 8.0.22 to replace [**group\_replication\_ip\_whitelist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_whitelist). From MySQL 8.0.24, the value of this system variable can be changed while Group Replication is running, and the change takes effect immediately on the member.

[**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) specifies which hosts are permitted to connect to the group. The address that you specify for each group member in [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) must be permitted on the other servers in the replication group. Note that the value you specify for this variable is not validated until a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement is issued and the Group Communication System (GCS) is available.

By default, this system variable is set to **AUTOMATIC**, which permits connections from private subnetworks active on the host. The group communication engine for Group Replication (XCom) automatically scans active interfaces on the host, and identifies those with addresses on private subnetworks. These addresses and the **localhost** IP address for IPv4 and (from MySQL 8.0.14) IPv6 are used to create the Group Replication allowlist. For a list of the ranges from which addresses are automatically permitted, see [Section 18.6.1, “Group Replication IP Address Permissions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-ip-address-permissions).

The automatic allowlist of private addresses cannot be used for connections from servers outside the private network. For Group Replication connections between server instances that are on different machines, you must provide public IP addresses and specify these as an explicit allowlist. If you specify any entries for the allowlist, the private addresses are not added automatically, so if you use any of these, you must specify them explicitly. The **localhost** IP addresses are added automatically.

As the value of the [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) option, you can specify any combination of the following:

IPv4 addresses (for example, **198.51.100.44**)

IPv4 addresses with CIDR notation (for example, **192.0.2.21/24**)

IPv6 addresses, from MySQL 8.0.14 (for example, **2001:db8:85a3:8d3:1319:8a2e:370:7348**)

IPv6 addresses with CIDR notation, from MySQL 8.0.14 (for example, **2001:db8:85a3:8d3::/64**)

Host names (for example, **example.org**)

Host names with CIDR notation (for example, **www.example.com/24**)

Before MySQL 8.0.14, host names could only resolve to IPv4 addresses. From MySQL 8.0.14, host names can resolve to IPv4 addresses, IPv6 addresses, or both. If a host name resolves to both an IPv4 and an IPv6 address, the IPv4 address is always used for Group Replication connections. You can use CIDR notation in combination with host names or IP addresses to permit a block of IP addresses with a particular network prefix, but do ensure that all the IP addresses in the specified subnet are under your control.

A comma must separate each entry in the allowlist. For example:

"192.0.2.21/24,198.51.100.44,203.0.113.0/24,2001:db8:85a3:8d3:1319:8a2e:370:7348,example.org,www.example.com/24"

If any of the seed members for the group are listed in the [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) option with an IPv6 address when a joining member has an IPv4 [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address), or the reverse, you must also set up and permit an alternative address for the joining member for the protocol offered by the seed member (or a host name that resolves to an address for that protocol). For more information, see [Section 18.6.1, “Group Replication IP Address Permissions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-ip-address-permissions).

It is possible to configure different allowlists on different group members according to your security requirements, for example, in order to keep different subnets separate. However, this can cause issues when a group is reconfigured. If you do not have a specific security requirement to do otherwise, use the same allowlist on all members of a group. For more details, see [Section 18.6.1, “Group Replication IP Address Permissions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-ip-address-permissions).

For host names, name resolution takes place only when a connection request is made by another server. A host name that cannot be resolved is not considered for allowlist validation, and a warning message is written to the error log. Forward-confirmed reverse DNS (FCrDNS) verification is carried out for resolved host names.

**Warning**

Host names are inherently less secure than IP addresses in an allowlist. FCrDNS verification provides a good level of protection, but can be compromised by certain types of attack. Specify host names in your allowlist only when strictly necessary, and ensure that all components used for name resolution, such as DNS servers, are maintained under your control. You can also implement name resolution locally using the hosts file, to avoid the use of external components.

**[group\_replication\_ip\_whitelist](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_ip_whitelist)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-ip-whitelist=value** |
| **Deprecated** | 8.0.22 |
| **System Variable** | [**group\_replication\_ip\_whitelist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_whitelist) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **AUTOMATIC** |

From MySQL 8.0.22, [**group\_replication\_ip\_whitelist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_whitelist) is deprecated, and [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) is available to replace it. For both system variables, the default value is **AUTOMATIC**.

At Group Replication startup, if either one of the system variables has been set to a user-defined value and the other has not, the changed value is used. If both of the system variables have been set to a user-defined value, the value of [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) is used.

If you change the value of [**group\_replication\_ip\_whitelist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_whitelist) or [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) while Group Replication is running, which is possible from MySQL 8.0.24, neither variable has precedence over the other.

The new system variable works in the same way as the old system variable, only the terminology has changed. The behavior description given for [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) applies to both the old and new system variables.

**[group\_replication\_local\_address](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_local_address)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-local-address=value** |
| **System Variable** | [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) sets the network address which the member provides for connections from other members, specified as a **host:port** formatted string. This address must be reachable by all members of the group because it is used by the group communication engine for Group Replication (XCom, a Paxos variant) for TCP communication between remote XCom instances. Communication with the local instance is over an input channel using shared memory.

**Warning**

Do not use this address for communication with the member. This is not the MySQL server SQL protocol host and port.

The address or host name that you specify in [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) is used by Group Replication as the unique identifier for a group member within the replication group. You can use the same port for all members of a replication group as long as the host names or IP addresses are all different, and you can use the same host name or IP address for all members as long as the ports are all different. The recommended port for [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) is 33061. Note that the value you specify for this variable is not validated until the [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement is issued and the Group Communication System (GCS) is available.

The network address configured by [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) must be resolvable by all group members. For example, if each server instance is on a different machine with a fixed network address, you could use the IP address of the machine, such as 10.0.0.1. If you use a host name, you must use a fully qualified name, and ensure it is resolvable through DNS, correctly configured **/etc/hosts** files, or other name resolution processes. From MySQL 8.0.14, IPv6 addresses (or host names that resolve to them) can be used as well as IPv4 addresses. An IPv6 address must be specified in square brackets in order to distinguish the port number, for example:

group\_replication\_local\_address= "[2001:db8:85a3:8d3:1319:8a2e:370:7348]:33061"

If a host name specified as the Group Replication local address for a server instance resolves to both an IPv4 and an IPv6 address, the IPv4 address is always used for Group Replication connections. For more information on Group Replication support for IPv6 networks and on replication groups with a mix of members using IPv4 and members using IPv6, see [Section 18.5.5, “Support For IPv6 And For Mixed IPv6 And IPv4 Groups”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-ipv6).

For the purpose of IP address permissions for Group Replication, the address that you specify for each group member in [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) must be added to the list for the [**group\_replication\_ip\_allowlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_allowlist) (from MySQL 8.0.22) or [**group\_replication\_ip\_whitelist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ip_whitelist) system variable on the other servers in the replication group. If any of the seed members for the group are listed in the [**group\_replication\_group\_seeds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_seeds) option with an IPv6 address when this member has an IPv4 [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address), or the reverse, you must also set up and permit an alternative address for this member for the required protocol (or a host name that resolves to an address for that protocol). For more information, see [Section 18.6.1, “Group Replication IP Address Permissions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-ip-address-permissions).

**[group\_replication\_member\_expel\_timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_member_expel_timeout)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-member-expel-timeout=#** |
| **Introduced** | 8.0.13 |
| **System Variable** | [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value (≥ 8.0.21)** | **5** |
| **Default Value (≤ 8.0.20)** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value (≥ 8.0.14)** | **3600** |
| **Maximum Value (≤ 8.0.13)** | **31536000** |
| **Unit** | **seconds** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately. The current value of the system variable is read whenever Group Replication checks the timeout. It is not mandatory for all members of a group to have the same setting, but it is recommended in order to avoid unexpected expulsions.

[**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) specifies the period of time in seconds that a Group Replication group member waits after creating a suspicion, before expelling from the group the member suspected of having failed. The initial 5-second detection period before a suspicion is created does not count as part of this time. Up to and including MySQL 8.0.20, the value of [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) defaults to 0, meaning that there is no waiting period and a suspected member is liable for expulsion immediately after the 5-second detection period ends. From MySQL 8.0.21, the value defaults to 5, meaning that a suspected member is liable for expulsion 5 seconds after the 5-second detection period.

Changing the value of [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) on a group member takes effect immediately for existing as well as future suspicions on that group member. You can therefore use this as a method to force a suspicion to time out and expel a suspected member, allowing changes to the group configuration. For more information, see [Section 18.7.6.1, “Expel Timeout”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-responses-failure-expel).

Increasing the value of [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) can help to avoid unnecessary expulsions on slower or less stable networks, or in the case of expected transient network outages or machine slowdowns. If a suspect member becomes active again before the suspicion times out, it applies all the messages that were buffered by the remaining group members and enters **ONLINE** state, without operator intervention. You can specify a timeout value up to a maximum of 3600 seconds (1 hour). It is important to ensure that XCom's message cache is sufficiently large to contain the expected volume of messages in your specified time period, plus the initial 5-second detection period, otherwise members are unable to reconnect. You can adjust the cache size limit using the [**group\_replication\_message\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_message_cache_size) system variable. For more information, see [Section 18.7.5, “XCom Cache Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-performance-xcom-cache).

If the timeout is exceeded, the suspect member is liable for expulsion immediately after the suspicion times out. If the member is able to resume communications and receives a view where it is expelled, and the member has the [**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries) system variable set to specify a number of auto-rejoin attempts, it proceeds to make the specified number of attempts to rejoin the group while in super read only mode. If the member does not have any auto-rejoin attempts specified, or if it has exhausted the specified number of attempts, it follows the action specified by the system variable [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action).

For more information on using the [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) setting, see [Section 18.7.6.1, “Expel Timeout”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-responses-failure-expel). For alternative mitigation strategies to avoid unnecessary expulsions where this system variable is not available, see [Section 18.3.2, “Group Replication Limitations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-limitations).

**[group\_replication\_member\_weight](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_member_weight)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-member-weight=#** |
| **System Variable** | [**group\_replication\_member\_weight**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_weight) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **50** |
| **Minimum Value** | **0** |
| **Maximum Value** | **100** |
| **Unit** | **percentage** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately. The system variable's current value is read when a failover situation occurs.

[**group\_replication\_member\_weight**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_weight) specifies a percentage weight that can be assigned to members to influence the chance of the member being elected as primary in the event of failover, for example when the existing primary leaves a single-primary group. Assign numeric weights to members to ensure that specific members are elected, for example during scheduled maintenance of the primary or to ensure certain hardware is prioritized in the event of failover.

For a group with members configured as follows:

**member-1**: group\_replication\_member\_weight=30, server\_uuid=aaaa

**member-2**: group\_replication\_member\_weight=40, server\_uuid=bbbb

**member-3**: group\_replication\_member\_weight=40, server\_uuid=cccc

**member-4**: group\_replication\_member\_weight=40, server\_uuid=dddd

during election of a new primary the members above would be sorted as **member-2**, **member-3**, **member-4**, and **member-1**. This results in **member**-2 being chosen as the new primary in the event of failover. For more information, see [Section 18.1.3.1, “Single-Primary Mode”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-single-primary-mode).

**[group\_replication\_message\_cache\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_message_cache_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-message-cache-size=#** |
| **Introduced** | 8.0.16 |
| **System Variable** | [**group\_replication\_message\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_message_cache_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1073741824 (1 GB)** |
| **Minimum Value (64-bit platforms, ≥ 8.0.21)** | **134217728 (128 MB)** |
| **Minimum Value (64-bit platforms, ≤ 8.0.20)** | **1073741824 (1 GB)** |
| **Minimum Value (32-bit platforms, ≥ 8.0.21)** | **134217728 (128 MB)** |
| **Minimum Value (32-bit platforms, ≤ 8.0.20)** | **1073741824 (1 GB)** |
| **Maximum Value (64-bit platforms)** | **18446744073709551615 (16 EiB)** |
| **Maximum Value (32-bit platforms)** | **315360004294967295 (4 GB)** |
| **Unit** | **bytes** |

This system variable should have the same value on all group members. The value of this system variable can be changed while Group Replication is running. The change takes effect on each group member after you stop and restart Group Replication on the member. During this process, the value of the system variable is permitted to differ between group members, but members might be unable to reconnect in the event of a disconnection.

[**group\_replication\_message\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_message_cache_size) sets the maximum amount of memory that is available for the message cache in the group communication engine for Group Replication (XCom). The XCom message cache holds messages (and their metadata) that are exchanged between the group members as a part of the consensus protocol. Among other functions, the message cache is used for recovery of missed messages by members that reconnect with the group after a period where they were unable to communicate with the other group members.

The [**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) system variable determines the waiting period (up to an hour) that is allowed in addition to the initial 5-second detection period for members to return to the group rather than being expelled. The size of the XCom message cache should be set with reference to the expected volume of messages in this time period, so that it contains all the missed messages required for members to return successfully. Up to MySQL 8.0.20, the default is only the 5-second detection period, but from MySQL 8.0.21, the default is a 5-second waiting period after the 5-second detection period, for a total time period of 10 seconds.

Ensure that sufficient memory is available on your system for your chosen cache size limit, considering the size of MySQL Server's other caches and object pools. The default setting is 1073741824 bytes (1 GB). The minimum setting is also 1 GB up to MySQL 8.0.20. From MySQL 8.0.21, the minimum setting is 134217728 bytes (128 MB), which enables deployment on a host that has a restricted amount of available memory, and good network connectivity to minimize the frequency and duration of transient losses of connectivity for group members. Note that the limit set using [**group\_replication\_message\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_message_cache_size) applies only to the data stored in the cache, and the cache structures require an additional 50 MB of memory.

The cache size limit can be increased or reduced dynamically at runtime. If you reduce the cache size limit, XCom removes the oldest entries that have been decided and delivered until the current size is below the limit. Group Replication's Group Communication System (GCS) alerts you, by a warning message, when a message that is likely to be needed for recovery by a member that is currently unreachable is removed from the message cache. For more information on tuning the message cache size, see [Section 18.7.5, “XCom Cache Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-performance-xcom-cache).

**[group\_replication\_poll\_spin\_loops](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_poll_spin_loops)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-poll-spin-loops=#** |
| **System Variable** | [**group\_replication\_poll\_spin\_loops**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_poll_spin_loops) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value (64-bit platforms)** | **18446744073709551615** |
| **Maximum Value (32-bit platforms)** | **4294967295** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_poll\_spin\_loops**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_poll_spin_loops) specifies the number of times the group communication thread waits for the communication engine mutex to be released before the thread waits for more incoming network messages.

**[group\_replication\_recovery\_complete\_at](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_complete_at)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-complete-at=value** |
| **System Variable** | [**group\_replication\_recovery\_complete\_at**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_complete_at) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **TRANSACTIONS\_APPLIED** |
| **Valid Values** | **TRANSACTIONS\_CERTIFIED**  **TRANSACTIONS\_APPLIED** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_complete\_at**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_complete_at) specifies the policy applied during the distributed recovery process when handling cached transactions after state transfer from an existing member. You can choose whether a member is marked online after it has received and certified all transactions that it missed before it joined the group (**TRANSACTIONS\_CERTIFIED**), or only after it has received, certified, and applied them (**TRANSACTIONS\_APPLIED**).

**[group\_replication\_recovery\_compression\_algorithms](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_compression_algorithms)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-compression-algorithms=value** |
| **Introduced** | 8.0.18 |
| **System Variable** | [**group\_replication\_recovery\_compression\_algorithms**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_compression_algorithms) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Set |
| **Default Value** | **uncompressed** |
| **Valid Values** | **zlib**  **zstd**  **uncompressed** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_compression\_algorithms**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_compression_algorithms) specifies the compression algorithms permitted for Group Replication distributed recovery connections for state transfer from a donor's binary log. The available algorithms are the same as for the [**protocol\_compression\_algorithms**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_protocol_compression_algorithms) system variable. For more information, see [Section 4.2.8, “Connection Compression Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#connection-compression-control).

This setting does not apply if the server has been set up to support cloning (see [Section 18.5.3.2, “Cloning for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-cloning)) and a remote cloning operation is used during distributed recovery. For this method of state transfer, the clone plugin's [**clone\_enable\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_enable_compression) setting applies.

**[group\_replication\_recovery\_get\_public\_key](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_get_public_key)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-get-public-key[={OFF|ON}]** |
| **System Variable** | [**group\_replication\_recovery\_get\_public\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_get_public_key) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_get\_public\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_get_public_key) specifies whether to request from the source the public key required for RSA key pair-based password exchange. If [**group\_replication\_recovery\_public\_key\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_public_key_path) is set to a valid public key file, it takes precedence over [**group\_replication\_recovery\_get\_public\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_get_public_key). This variable applies if you are not using SSL for distributed recovery over the **group\_replication\_recovery** channel ([**group\_replication\_recovery\_use\_ssl=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_use_ssl)), and the replication user account for Group Replication authenticates with the **caching\_sha2\_password** plugin (which is the default in MySQL 8.0). For more details, see [Section 18.6.3.1.1, “Replication User With The Caching SHA-2 Authentication Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-caching-sha2-user-credentials).

**[group\_replication\_recovery\_public\_key\_path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_public_key_path)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-public-key-path=file\_name** |
| **System Variable** | [**group\_replication\_recovery\_public\_key\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_public_key_path) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | File name |
| **Default Value** | **NULL** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_public\_key\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_public_key_path) specifies the path name to a file containing a replica-side copy of the public key required by the source for RSA key pair-based password exchange. The file must be in PEM format. If [**group\_replication\_recovery\_public\_key\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_public_key_path) is set to a valid public key file, it takes precedence over [**group\_replication\_recovery\_get\_public\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_get_public_key). This variable applies if you are not using SSL for distributed recovery over the **group\_replication\_recovery** channel (so [**group\_replication\_recovery\_use\_ssl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_use_ssl) is set to **OFF**), and the replication user account for Group Replication authenticates with the **caching\_sha2\_password** plugin (which is the default in MySQL 8.0) or the **sha256\_password** plugin. (For **sha256\_password**, setting **group\_replication\_recovery\_public\_key\_path** applies only if MySQL was built using OpenSSL.) For more details, see [Section 18.6.3.1.1, “Replication User With The Caching SHA-2 Authentication Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-caching-sha2-user-credentials).

**[group\_replication\_recovery\_reconnect\_interval](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_reconnect_interval)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-reconnect-interval=#** |
| **System Variable** | [**group\_replication\_recovery\_reconnect\_interval**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_reconnect_interval) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **60** |
| **Minimum Value** | **0** |
| **Maximum Value** | **31536000** |
| **Unit** | **seconds** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_reconnect\_interval**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_reconnect_interval) specifies the sleep time, in seconds, between reconnection attempts when no suitable donor was found in the group for distributed recovery.

**[group\_replication\_recovery\_retry\_count](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_retry_count)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-retry-count=#** |
| **System Variable** | [**group\_replication\_recovery\_retry\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_retry_count) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **10** |
| **Minimum Value** | **0** |
| **Maximum Value** | **31536000** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_retry\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_retry_count) specifies the number of times that the member that is joining tries to connect to the available donors for distributed recovery before giving up.

**[group\_replication\_recovery\_ssl\_ca](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_ssl_ca)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-ssl-ca=value** |
| **System Variable** | [**group\_replication\_recovery\_ssl\_ca**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_ca) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_ssl\_ca**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_ca) specifies the path to a file that contains a list of trusted SSL certificate authorities for distributed recovery connections. See [Section 18.6.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl) for information on configuring SSL for distributed recovery.

If this server has been set up to support cloning (see [Section 18.5.3.2, “Cloning for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-cloning)), and you have set [**group\_replication\_recovery\_use\_ssl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_use_ssl) to **ON**, Group Replication automatically configures the setting for the clone SSL option [**clone\_ssl\_ca**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_ssl_ca) to match your setting for [**group\_replication\_recovery\_ssl\_ca**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_ca).

**[group\_replication\_recovery\_ssl\_capath](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_ssl_capath)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-ssl-capath=value** |
| **System Variable** | [**group\_replication\_recovery\_ssl\_capath**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_capath) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_ssl\_capath**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_capath) specifies the path to a directory that contains trusted SSL certificate authority certificates for distributed recovery connections. See [Section 18.6.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl) for information on configuring SSL for distributed recovery.

**[group\_replication\_recovery\_ssl\_cert](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_ssl_cert)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-ssl-cert=value** |
| **System Variable** | [**group\_replication\_recovery\_ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_cert) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_cert) specifies the name of the SSL certificate file to use for establishing a secure connection for distributed recovery. See [Section 18.6.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl) for information on configuring SSL for distributed recovery.

If this server has been set up to support cloning (see [Section 18.5.3.2, “Cloning for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-cloning)), and you have set [**group\_replication\_recovery\_use\_ssl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_use_ssl) to **ON**, Group Replication automatically configures the setting for the clone SSL option [**clone\_ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_ssl_cert) to match your setting for [**group\_replication\_recovery\_ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_cert).

**[group\_replication\_recovery\_ssl\_cipher](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_ssl_cipher)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-ssl-cipher=value** |
| **System Variable** | [**group\_replication\_recovery\_ssl\_cipher**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_cipher) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_ssl\_cipher**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_cipher) specifies the list of permissible ciphers for SSL encryption. See [Section 18.6.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl) for information on configuring SSL for distributed recovery.

**[group\_replication\_recovery\_ssl\_crl](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_ssl_crl)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-ssl-crl=value** |
| **System Variable** | [**group\_replication\_recovery\_ssl\_crl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_crl) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | File name |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_ssl\_crl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_crl) specifies the path to a directory that contains files containing certificate revocation lists. See [Section 18.6.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl) for information on configuring SSL for distributed recovery.

**[group\_replication\_recovery\_ssl\_crlpath](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_ssl_crlpath)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-ssl-crlpath=value** |
| **System Variable** | [**group\_replication\_recovery\_ssl\_crlpath**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_crlpath) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Directory name |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_ssl\_crlpath**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_crlpath) specifies the path to a directory that contains files containing certificate revocation lists. See [Section 18.6.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl) for information on configuring SSL for distributed recovery.

**[group\_replication\_recovery\_ssl\_key](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_ssl_key)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-ssl-key=value** |
| **System Variable** | [**group\_replication\_recovery\_ssl\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_key) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_ssl\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_key) specifies the name of the SSL key file to use for establishing a secure connection. See [Section 18.6.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl) for information on configuring SSL for distributed recovery.

If this server has been set up to support cloning (see [Section 18.5.3.2, “Cloning for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-cloning)), and you have set [**group\_replication\_recovery\_use\_ssl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_use_ssl) to **ON**, Group Replication automatically configures the setting for the clone SSL option [**clone\_ssl\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_clone_ssl_key) to match your setting for [**group\_replication\_recovery\_ssl\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_key).

**[group\_replication\_recovery\_ssl\_verify\_server\_cert](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_ssl_verify_server_cert)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-ssl-verify-server-cert[={OFF|ON}]** |
| **System Variable** | [**group\_replication\_recovery\_ssl\_verify\_server\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_verify_server_cert) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_ssl\_verify\_server\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_ssl_verify_server_cert) specifies whether the distributed recovery connection should check the server's Common Name value in the certificate sent by the donor. See [Section 18.6.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl) for information on configuring SSL for distributed recovery.

**[group\_replication\_recovery\_tls\_ciphersuites](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_tls_ciphersuites)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-tls-ciphersuites=value** |
| **Introduced** | 8.0.19 |
| **System Variable** | [**group\_replication\_recovery\_tls\_ciphersuites**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_tls_ciphersuites) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **NULL** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_tls\_ciphersuites**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_tls_ciphersuites) specifies a colon-separated list of one or more permitted ciphersuites when TLSv1.3 is used for connection encryption for the distributed recovery connection, and this server instance is the client in the distributed recovery connection, that is, the joining member. If this system variable is set to **NULL** when TLSv1.3 is used (which is the default if you do not set the system variable), the ciphersuites that are enabled by default are allowed, as listed in [Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#encrypted-connection-protocols-ciphers). If this system variable is set to the empty string, no cipher suites are allowed, and TLSv1.3 is therefore not used. This system variable is available beginning with MySQL 8.0.19. See [Section 18.6.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl), for information on configuring SSL for distributed recovery.

**[group\_replication\_recovery\_tls\_version](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_tls_version)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-tls-version=value** |
| **Introduced** | 8.0.19 |
| **System Variable** | [**group\_replication\_recovery\_tls\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_tls_version) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **TLSv1,TLSv1.1,TLSv1.2,TLSv1.3** (OpenSSL 1.1.1 or higher)  **TLSv1,TLSv1.1,TLSv1.2** (otherwise) |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_tls\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_tls_version) specifies a comma-separated list of one or more permitted TLS protocols for connection encryption when this server instance is the client in the distributed recovery connection, that is, the joining member. Ensure the specified versions are contiguous (for example, “**TLSv1,TLSv1.1,TLSv1.2**”). If this system variable is not set, the default “**TLSv1,TLSv1.1,TLSv1.2,TLSv1.3**” is used. The group members involved in each distributed recovery connection as the client (joining member) and server (donor) negotiate the highest protocol version that they are both set up to support. This system variable is available from MySQL 8.0.19. See [Section 18.6.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl) for information on configuring SSL for distributed recovery.

**[group\_replication\_recovery\_use\_ssl](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_use_ssl)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-use-ssl[={OFF|ON}]** |
| **System Variable** | [**group\_replication\_recovery\_use\_ssl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_use_ssl) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_use\_ssl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_use_ssl) specifies whether Group Replication distributed recovery connections between group members should use SSL or not. See [Section 18.6.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl) for information on configuring SSL for distributed recovery.

If this server has been set up to support cloning (see [Section 18.5.3.2, “Cloning for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-cloning)), and you set this option to **ON**, Group Replication uses SSL for remote cloning operations as well as for state transfer from a donor's binary log. If you set this option to **OFF**, Group Replication does not use SSL for remote cloning operations.

**[group\_replication\_recovery\_zstd\_compression\_level](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_recovery_zstd_compression_level)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-recovery-zstd-compression-level=#** |
| **Introduced** | 8.0.18 |
| **System Variable** | [**group\_replication\_recovery\_zstd\_compression\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_zstd_compression_level) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **3** |
| **Minimum Value** | **1** |
| **Maximum Value** | **22** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_recovery\_zstd\_compression\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_recovery_zstd_compression_level) specifies the compression level to use for Group Replication distributed recovery connections that use the **zstd** compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default **zstd** compression level is 3. For distributed recovery connections that do not use **zstd** compression, this variable has no effect.

For more information, see [Section 4.2.8, “Connection Compression Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#connection-compression-control).

**[group\_replication\_single\_primary\_mode](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_single_primary_mode)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-single-primary-mode[={OFF|ON}]** |
| **System Variable** | [**group\_replication\_single\_primary\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_single_primary_mode) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

This system variable is a group-wide configuration setting. It must have the same value on all group members, cannot be changed while Group Replication is running, and requires a full reboot of the group (a bootstrap by a server with [**group\_replication\_bootstrap\_group=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_bootstrap_group)) in order for the value change to take effect. From MySQL 8.0.16, you can use the [**group\_replication\_switch\_to\_single\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-single-primary-mode) and [**group\_replication\_switch\_to\_multi\_primary\_mode()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-switch-to-multi-primary-mode) UDFs to change the value of this system variable while the group is still running. For more information, see [Section 18.5.1.2, “Changing a Group's Mode”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-changing-group-mode).

[**group\_replication\_single\_primary\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_single_primary_mode) instructs the group to automatically pick a single server to be the one that handles read/write workload. This server is the PRIMARY and all others are SECONDARIES.

**[group\_replication\_ssl\_mode](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_ssl_mode)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-ssl-mode=value** |
| **System Variable** | [**group\_replication\_ssl\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ssl_mode) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **DISABLED** |
| **Valid Values** | **DISABLED**  **REQUIRED**  **VERIFY\_CA**  **VERIFY\_IDENTITY** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_ssl\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_ssl_mode) sets the security state of group communication connections between Group Replication members. The possible values are as follows:

**DISABLED**

Establish an unencrypted connection (the default).

**REQUIRED**

Establish a secure connection if the server supports secure connections.

**VERIFY\_CA**

Like **REQUIRED**, but additionally verify the server TLS certificate against the configured Certificate Authority (CA) certificates.

**VERIFY\_IDENTITY**

Like **VERIFY\_CA**, but additionally verify that the server certificate matches the host to which the connection is attempted.

See [Section 18.6.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-socket-layer-support-ssl) for information on configuring SSL for group communication.

**[group\_replication\_start\_on\_boot](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_start_on_boot)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-start-on-boot[={OFF|ON}]** |
| **System Variable** | [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) specifies whether the server should start Group Replication automatically (**ON**) or not (**OFF**) during server start. When you set this option to **ON**, Group Replication restarts automatically after a remote cloning operation is used for distributed recovery.

To start Group Replication automatically during server start, the user credentials for distributed recovery must be stored in the replication metadata repositories on the server using the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement. If you prefer to specify the user credentials on the [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement, which stores the user credentials only in memory, ensure that [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) is set to **OFF**.

**[group\_replication\_tls\_source](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_tls_source)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-tls-source=value** |
| **Introduced** | 8.0.21 |
| **System Variable** | [**group\_replication\_tls\_source**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_tls_source) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **mysql\_main** |
| **Valid Values** | **mysql\_main**  **mysql\_admin** |

The value of this system variable can be changed while Group Replication is running, but the change only takes effect after you stop and restart Group Replication on the group member.

[**group\_replication\_tls\_source**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_tls_source) specifies the source of TLS material for Group Replication.

**[group\_replication\_transaction\_size\_limit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_transaction_size_limit)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-transaction-size-limit=#** |
| **System Variable** | [**group\_replication\_transaction\_size\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_transaction_size_limit) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **150000000** |
| **Minimum Value** | **0** |
| **Maximum Value** | **2147483647** |
| **Unit** | **bytes** |

This system variable should have the same value on all group members. The value of this system variable can be changed while Group Replication is running. The change takes effect immediately on the group member, and applies from the next transaction started on that member. During this process, the value of the system variable is permitted to differ between group members, but some transactions might be rejected.

[**group\_replication\_transaction\_size\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_transaction_size_limit) configures the maximum transaction size in bytes which the replication group accepts. Transactions larger than this size are rolled back by the receiving member and are not broadcast to the group. Large transactions can cause problems for a replication group in terms of memory allocation, which can cause the system to slow down, or in terms of network bandwidth consumption, which can cause a member to be suspected of having failed because it is busy processing the large transaction.

When this system variable is set to 0 there is no limit to the size of transactions the group accepts. From MySQL 8.0, the default setting for this system variable is 150000000 bytes (approximately 143 MB). Adjust the value of this system variable depending on the maximum message size that you need the group to tolerate, bearing in mind that the time taken to process a transaction is proportional to its size. The value of [**group\_replication\_transaction\_size\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_transaction_size_limit) should be the same on all group members. For further mitigation strategies for large transactions, see [Section 18.3.2, “Group Replication Limitations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-limitations).

**[group\_replication\_unreachable\_majority\_timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\group-replication.html" \l "sysvar_group_replication_unreachable_majority_timeout)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--group-replication-unreachable-majority-timeout=#** |
| **System Variable** | [**group\_replication\_unreachable\_majority\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_unreachable_majority_timeout) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **31536000** |
| **Unit** | **seconds** |

The value of this system variable can be changed while Group Replication is running, and the change takes effect immediately. The current value of the system variable is read when an issue occurs that means the behavior is needed.

[**group\_replication\_unreachable\_majority\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_unreachable_majority_timeout) specifies a number of seconds for which members that suffer a network partition and cannot connect to the majority wait before leaving the group. In a group of 5 servers (S1,S2,S3,S4,S5), if there is a disconnection between (S1,S2) and (S3,S4,S5) there is a network partition. The first group (S1,S2) is now in a minority because it cannot contact more than half of the group. While the majority group (S3,S4,S5) remains running, the minority group waits for the specified time for a network reconnection. For a detailed description of this scenario, see [Section 18.5.4, “Network Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-network-partitioning).

By default, [**group\_replication\_unreachable\_majority\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_unreachable_majority_timeout) is set to 0, which means that members that find themselves in a minority due to a network partition wait forever to leave the group. If you set a timeout, when the specified time elapses, all pending transactions processed by the minority are rolled back, and the servers in the minority partition move to the **ERROR** state. If a member has the [**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries) system variable set to specify a number of auto-rejoin attempts, it proceeds to make the specified number of attempts to rejoin the group while in super read only mode. If the member does not have any auto-rejoin attempts specified, or if it has exhausted the specified number of attempts, it follows the action specified by the system variable [**group\_replication\_exit\_state\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_exit_state_action).

**Warning**

When you have a symmetric group, with just two members for example (S0,S2), if there is a network partition and there is no majority, after the configured timeout all members enter the **ERROR** state.

For more information on using this option, see [Section 18.7.6.2, “Unreachable Majority Timeout”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-responses-failure-partition).

### Group Replication Status Variable

This section describes the status variable which provides information about Group Replication. The variable has the following meaning:

[**group\_replication\_primary\_member**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_group_replication_primary_member)

Shows the primary member's UUID when the group is operating in single-primary mode. If the group is operating in multi-primary mode, shows an empty string.

**Warning**

The **group\_replication\_primary\_member** status variable has been deprecated and is scheduled to be removed in a future version.

See [Section 18.1.3.1.2, “Finding the Primary”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-find-primary).

## 18.10 Frequently Asked Questions

This section provides answers to frequently asked questions.

### What is the maximum number of MySQL servers in a group?

A group can consist of maximum 9 servers. Attempting to add another server to a group with 9 members causes the request to join to be refused. This limit has been identified from testing and benchmarking as a safe boundary where the group performs reliably on a stable local area network.

### How are servers in a group connected?

Servers in a group connect to the other servers in the group by opening a peer-to-peer TCP connection. These connections are only used for internal communication and message passing between servers in the group. This address is configured by the [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address) variable.

### What is the group\_replication\_bootstrap\_group option used for?

The bootstrap flag instructs a member to create a group and act as the initial seed server. The second member joining the group needs to ask the member that bootstrapped the group to dynamically change the configuration in order for it to be added to the group.

A member needs to bootstrap the group in two scenarios. When the group is originally created, or when shutting down and restarting the entire group.

### How do I set credentials for the distributed recovery process?

You can set the user credentials permanently as the credentials for the **group\_replication\_recovery** channel, using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). Alternatively, from MySQL 8.0.21, you can specify them on the [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement each time Group Replication is started.

User credentials set using [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) are stored in plain text in the replication metadata repositories on the server, but user credentials specified on [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) are saved in memory only, and are removed by a [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) statement or server shutdown. Using [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) to specify the user credentials therefore helps to secure the Group Replication servers against unauthorized access. However, this method is not compatible with starting Group Replication automatically, as specified by the [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) system variable. For more information, see [Section 18.6.3.1, “Secure User Credentials for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-secure-user).

### Can I scale-out my write-load using Group Replication?

Not directly, but MySQL Group replication is a shared nothing full replication solution, where all servers in the group replicate the same amount of data. Therefore if one member in the group writes N bytes to storage as the result of a transaction commit operation, then roughly N bytes are written to storage on other members as well, because the transaction is replicated everywhere.

However, given that other members do not have to do the same amount of processing that the original member had to do when it originally executed the transaction, they apply the changes faster. Transactions are replicated in a format that is used to apply row transformations only, without having to re-execute transactions again (row-based format).

Furthermore, given that changes are propagated and applied in row-based format, this means that they are received in an optimized and compact format, and likely reducing the number of IO operations required when compared to the originating member.

To summarize, you can scale-out processing, by spreading conflict free transactions throughout different members in the group. And you can likely scale-out a small fraction of your IO operations, since remote servers receive only the necessary changes to read-modify-write changes to stable storage.

### Does Group Replication require more network bandwidth and CPU, when compared to simple replication and under the same workload?

Some additional load is expected because servers need to be constantly interacting with each other for synchronization purposes. It is difficult to quantify how much more data. It also depends on the size of the group (three servers puts less stress on the bandwidth requirements than nine servers in the group).

Also the memory and CPU footprint are larger, because more complex work is done for the server synchronization part and for the group messaging.

### Can I deploy Group Replication across wide-area networks?

Yes, but the network connection between each member must be reliable and have suitable perfomance. Low latency, high bandwidth network connections are a requirement for optimal performance.

If network bandwidth alone is an issue, then [Section 18.7.3, “Message Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-message-compression) can be used to lower the bandwidth required. However, if the network drops packets, leading to re-transmissions and higher end-to-end latency, throughput and latency are both negatively affected.

**Warning**

When the network round-trip time (RTT) between any group members is 5 seconds or more you could encounter problems as the built-in failure detection mechanism could be incorrectly triggered.

### Do members automatically rejoin a group in case of temporary connectivity problems?

This depends on the reason for the connectivity problem. If the connectivity problem is transient and the reconnection is quick enough that the failure detector is not aware of it, then the server may not be removed from the group. If it is a "long" connectivity problem, then the failure detector eventually suspects a problem and the server is removed from the group.

From MySQL 8.0, two settings are available to increase the chances of a member remaining in or rejoining a group:

[**group\_replication\_member\_expel\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_member_expel_timeout) increases the time between the creation of a suspicion (which happens after an initial 5-second detection period) and the expulsion of the member. You can set a waiting period of up to 1 hour. From MySQL 8.0.21, a waiting period of 5 seconds is set by default.

[**group\_replication\_autorejoin\_tries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_autorejoin_tries) makes a member try to rejoin the group after an expulsion or unreachable majority timeout. The member makes the specified number of auto-rejoin attempts five minutes apart. From MySQL 8.0.21, this feature is activated by default and the member makes three auto-rejoin attempts.

If a server is expelled from the group and any auto-rejoin attempts do not succeed, you need to join it back again. In other words, after a server is removed explicitly from the group you need to rejoin it manually (or have a script doing it automatically).

### When is a member excluded from a group?

If the member becomes silent, the other members remove it from the group configuration. In practice this may happen when the member has crashed or there is a network disconnection.

The failure is detected after a given timeout elapses for a given member and a new configuration without the silent member in it is created.

### What happens when one node is significantly lagging behind?

There is no method for defining policies for when to expel members automatically from the group. You need to find out why a member is lagging behind and fix that or remove the member from the group. Otherwise, if the server is so slow that it triggers the flow control, then the entire group slows down as well. The flow control can be configured according to the your needs.

### Upon suspicion of a problem in the group, is there a special member responsible for triggering a reconfiguration?

No, there is no special member in the group in charge of triggering a reconfiguration.

Any member can suspect that there is a problem. All members need to (automatically) agree that a given member has failed. One member is in charge of expelling it from the group, by triggering a reconfiguration. Which member is responsible for expelling the member is not something you can control or set.

### Can I use Group Replication for sharding?

Group Replication is designed to provide highly available replica sets; data and writes are duplicated on each member in the group. For scaling beyond what a single system can provide, you need an orchestration and sharding framework built around a number of Group Replication sets, where each replica set maintains and manages a given shard or partition of your total dataset. This type of setup, often called a “sharded cluster”, allows you to scale reads and writes linearly and without limit.

### How do I use Group Replication with SELinux?

If SELinux is enabled, which you can verify using **sestatus -v**, then you need to enable the use of the Group Replication communication port. See [Setting the TCP Port Context for Group Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#selinux-context-group-replication-port).

### How do I use Group Replication with iptables?

If **iptables** is enabled, then you need to open up the Group Replication port for communication between the machines. To see the current rules in place on each machine, issue **iptables -L**. Assuming the port configured is 33061, enable communication over the necessary port by issuing **iptables -A INPUT -p tcp --dport 33061 -j ACCEPT**.

### How do I recover the relay log for a replication channel used by a group member?

The replication channels used by Group Replication behave in the same way as replication channels used in asynchronous source to replica replication, and as such rely on the relay log. In the event of a change of the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) variable, or when the option is not set and the host name changes, there is a chance of errors. See [Section 17.2.4.1, “The Relay Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs-relaylog) for a recovery procedure in this situation. Alternatively, another way of fixing the issue specifically in Group Replication is to issue a [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) statement and then a [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement to restart the instance. The Group Replication plugin creates the **group\_replication\_applier** channel again.

### Why does Group Replication use two bind addresses?

Group Replication uses two bind addresses in order to split network traffic between the SQL address, used by clients to communicate with the member, and the [**group\_replication\_local\_address**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address), used internally by the group members to communicate. For example, assume a server with two network interfaces assigned to the network addresses **203.0.113.1** and **198.51.100.179**. In such a situation you could use **203.0.113.1:33061** for the internal group network address by setting [**group\_replication\_local\_address=203.0.113.1:33061**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_local_address). Then you could use **198.51.100.179** for [**hostname**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_hostname) and **3306** for the [**port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_port). Client SQL applications would then connect to the member at **198.51.100.179:3306**. This enables you to configure different rules on the different networks. Similarly, the internal group communication can be separated from the network connection used for client applications, for increased security.

### How does Group Replication use network addresses and hostnames?

Group Replication uses network connections between members and therefore its functionality is directly impacted by how you configure hostnames and ports. For example, Group Replication's distributed recovery process creates a connection to an existing group member using the server's hostname and port. When a member joins a group it receives the group membership information, using the network address information that is listed at [**performance\_schema.replication\_group\_members**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-group-members-table). One of the members listed in that table is selected as the donor of the missing data from the group to the joining member.

This means that any value you configure using a hostname, such as the SQL network address or the group seeds address, must be a fully qualified name and resolvable by each member of the group. You can ensure this for example through DNS, or correctly configured /etc/hosts files, or other local processes. If a you want to configure the **MEMBER\_HOST** value on a server, specify it using the [--report-host](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_host) option on the server before joining it to the group.

**Important**

The assigned value is used directly and is not affected by the [**skip\_name\_resolve**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_skip_name_resolve) system variable.

To configure **MEMBER\_PORT** on a server, specify it using the [**report\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_port) system variable.

### Why did the auto increment setting on the server change?

When Group Replication is started on a server, the value of [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) is changed to the value of [**group\_replication\_auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_auto_increment_increment), which defaults to 7, and the value of [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) is changed to the server ID. The changes are reverted when Group Replication is stopped. These settings avoid the selection of duplicate auto-increment values for writes on group members, which causes rollback of transactions. The default auto increment value of 7 for Group Replication represents a balance between the number of usable values and the permitted maximum size of a replication group (9 members).

The changes are only made and reverted if [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) and [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) each have their default value of 1. If their values have already been modified from the default, Group Replication does not alter them. From MySQL 8.0, the system variables are also not modified when Group Replication is in single-primary mode, where only one server writes.

### How do I find the primary?

If the group is operating in single-primary mode, it can be useful to find out which member is the primary. See [Section 18.1.3.1.2, “Finding the Primary”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-find-primary)

|  |  |  |
| --- | --- | --- |
| [Prev](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html) | Up | [Next](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-shell-userguide.html) |
| Chapter 17 Replication | [Home](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\index.html) | Chapter 19 MySQL Shell |